SYSTEM V APPLICATION BINARY INTERFACE

Intel386™ Architecture Processor Supplement

Fourth Edition

Copyright © 1990-1996 The Santa Cruz Operation, Inc. All rights reserved.

Copyright © 1990-1992 AT&T. All rights reserved.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the copyright owner, The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, California, 95060, USA. Copyright infringement is a serious matter under the United States and foreign Copyright Laws.

Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc.

SCO® UnixWare® is commercial computer software and, together with any related documentation, is subject to the restrictions on US Government use as set forth below. If this procurement is for a DOD agency, the following DFAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If this procurement is for a civilian government agency, this FAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: This computer software is submitted with restricted rights under Government Contract No. _______, if appropriate). It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (g)(3)(i) of FAR Clause 52.227-14 alt III or as otherwise expressly stated in the contract. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If any copyrighted software accompanies this publication, it is licensed to the End User only for use in strict accordance with the End User License Agreement, which should be read carefully before commencing use of the software.

ACKNOWLEDGEMENTS

"We acknowledge the contributions of the 88OPEN Consortium Ltd., portions of whose *System V ABI Implementation Guide for the M88000 Processor* and the *System V ABI M88000 Processor Networking Supplement* have been incorporated in this section of the ABI with permission."

TRADEMARKS

SCO, the SCO logo, The Santa Cruz Operation, and UnixWare are trademarks or registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries. Intel386, Intel486, and Pentium are trademarks of Intel Corporation. UNIX is a registered trademark in the USA and other countries, licensed exclusively through X/Open Company Limited. Motif is a trademark of the Open Software Foundation, Inc. NeWS is a registered trademark of Sun Microsystems, Inc. X11 and X Window System are trademarks of Massachusetts Institute of Technology. All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners.

DRAFT COPY
March 19, 1997
File: abi_386/copyright (Delta 44.3)
386:adm.book:sum

Contents

Table of Contents

SOFTWARE INSTALLATION

LOW-LEVEL SYSTEM INFORMATION

Table of Contents INTRODUCTION

OBJECT FILES

	PROGRAM LOADING AND DYNAMIC LINKING LIBRARIES DEVELOPMENT ENVIRONMENT EXECUTION ENVIRONMENT Index	
1	INTRODUCTION The Intel386 Architecture and the System V ABI How to Use the Intel386 Architecture ABI Supplement	1-1 1-2
2	SOFTWARE INSTALLATION Software Distribution Formats	2-1
3	LOW-LEVEL SYSTEM INFORMATION Machine Interface Function Calling Sequence Operating System Interface Coding Examples	3-1 3-9 3-20 3-34

Table of Contents i

IN	Index Index	IN-1
8	EXECUTION ENVIRONMENT Application Environment	8-1
7	DEVELOPMENT ENVIRONMENT Development Commands Software Packaging Tools	7-1 7-2
6	LIBRARIES Shared Library Names C Library System Data Interfaces	6-1 6-2 6-5
5	PROGRAM LOADING AND DYNAMIC LINKING Program Loading Dynamic Linking	5-1 5-5
4	OBJECT FILES ELF Header Sections Symbol Table Relocation	4-1 4-2 4-3 4-4

ii Table of Contents

Figures and Tables

Figure 3-1: Scalar Types	3-2
Figure 3-2: Structure Smaller Than a Word	3-4
Figure 3-3: No Padding	3-4
Figure 3-4: Internal Padding	3-4
Figure 3-5: Internal and Tail Padding	3-5
Figure 3-6: union Allocation	3-5
Figure 3-7: Bit-Field Ranges	3-6
Figure 3-8: Bit Numbering	3-7
Figure 3-9: Right-to-Left Allocation	3-7
Figure 3-10: Boundary Alignment	3-7
Figure 3-11: Storage Unit Sharing	3-8
Figure 3-12: union Allocation	3-8
Figure 3-13: Unnamed Bit-Fields	3-8
Figure 3-14: Processor Registers	3-9
Figure 3-15: Standard Stack Frame	3-10
Figure 3-16: Function Prologue	3-13
Figure 3-17: Function Epilogue	3-14
Figure 3-18: Stack Contents for Functions Returning struct/union	3-15
Figure 3-19: Function Prologue (Returning struct/union)	3-16
Figure 3-20: Function Epilogue	3-16
Figure 3-21: Integral and Pointer Arguments	3-17
Figure 3-22: Floating-Point Arguments	3-18
Figure 3-23: Structure and Union Arguments	3-18
Figure 3-24: Virtual Address Configuration	3-20
Figure 3-25: Conventional Segment Arrangements	3-22
Figure 3-26: _exit System Trap	3-24
Figure 3-27: Hardware Exceptions and Signals	3-25
Figure 3-28: Declaration for main	3-26
Figure 3-29: EFLAGS Register Fields	3-27
Figure 3-30: Floating-Point Control Word	3-27
Figure 3-31: Initial Process Stack	3-28
Figure 3-32: Auxiliary Vector	3-30
Figure 3-33: Auxiliary Vector Types, a_type	3-30
Figure 3-34: AT_FPHW values	3-32
Figure 3-35: Example Process Stack	3-33

Table of Contents iii

Figure 3-36: Calculating Global Offset Table Address	3-36
Figure 3-37: Position-Independent Function Prologue	3-37
Figure 3-38: Absolute Data Access	3-37
Figure 3-39: Position-Independent Data Access	3-38
Figure 3-40: Position-Independent Static Data Access	3-39
Figure 3-41: Absolute Direct Function Call	3-39
Figure 3-42: Position-Independent Direct Function Call	3-40
Figure 3-43: Absolute Indirect Function Call	3-40
Figure 3-44: Position-Independent Indirect Function Call	3-40
Figure 3-45: Branch Instruction, All Models	3-41
Figure 3-46: Absolute switch Code	3-41
Figure 3-47: Position-Independent switch Code	3-42
Figure 3-48: C Stack Frame	3-43
Figure 3-49: Dynamic Stack Allocation	3-45
Figure 4-1: Intel386 Identification, e_ident	4-1
Figure 4-2: Special Sections	4-2
Figure 4-3: Relocatable Fields	4-4
Figure 4-4: Relocation Types	4-5
Figure 5-1: Executable File	5-1
Figure 5-2: Program Header Segments	5-2
Figure 5-3: Process Image Segments	5-3
Figure 5-4: Example Shared Object Segment Addresses	5-4
Figure 5-5: Global Offset Table	5-6
Figure 5-6: Absolute Procedure Linkage Table	5-8
Figure 5-7: Position-Independent Procedure Linkage Table	5-8
Figure 6-1: Shared Library Names	6-1
Figure 6-2: libc Additional Required Entry Points	6-2
Figure 6-3: libc, Support Routines	6-3
Figure 6-4: libc, Global External Data Symbols	6-4
Figure 6-5: <aio.h>*</aio.h>	6-6
Figure 6-6: <assert.h></assert.h>	6-6
Figure 6-7: <ctype.h></ctype.h>	6-7
Figure 6-8: <dirent.h></dirent.h>	6-8
Figure 6-9: <dlfcn.h>*</dlfcn.h>	6-8
Figure 6-10: <elf.h>*, Part 1 of 6</elf.h>	6-9
Figure 6-11: <elf.h>*, Part 2 of 6</elf.h>	6-10
Figure 6-12: <elf.h>*, Part 3 of 6</elf.h>	6-11
Figure 6-13: <elf.h>*, Part 4 of 6</elf.h>	6-12
Figure 6-14: <elf.h>*, Part 5 of 6</elf.h>	6-13
Figure 6-15: <elf.h>*, Part 6 of 6</elf.h>	6-14
Figure 6-16: <errno.h>, Part 1 of 3</errno.h>	6-15 6-16
Figure 6-17: <errno.h>, Part 2 of 3</errno.h>	6-16 6-17
Figure 6-18: <errno.h>, Part 3 of 3 Figure 6-19: <fant.l h="">, Part 1 of 2</fant.l></errno.h>	6-17 6-18
Figure 6-19: <fcntl.h>, Part 1 of 2</fcntl.h>	
Figure 6-20: <fcntl.h>, Part 2 of 2</fcntl.h>	6-19

iv Table of Contents

Figure 6-21:	<float.h>, Single-Precision</float.h>	6-20
Figure 6-22:	<float.h>, Double-Precision</float.h>	6-20
Figure 6-23:	<float.h>, Extended-Precision</float.h>	6-21
Figure 6-24:	<fmtmsg.h>, Part 1 of 2</fmtmsg.h>	6-21
Figure 6-25:	<fmtmsg.h>, Part 2 of 2</fmtmsg.h>	6-22
	<fnmatch.h>*</fnmatch.h>	6-22
Figure 6-27:	<ftw.h></ftw.h>	6-23
Figure 6-28:		6-24
Figure 6-29:	<grp.h></grp.h>	6-25
Figure 6-30:		6-25
Figure 6-31:	<sys ipc.h=""></sys>	6-26
Figure 6-32:	<pre><langinfo.h>, Part 1 of 2</langinfo.h></pre>	6-27
•	<pre><langinfo.h>, Part 2 of 2</langinfo.h></pre>	6-28
Figure 6-34:	<pre><limits.h>, Part 1 of 2</limits.h></pre>	6-29
Figure 6-35:	<pre><limits.h>, Part 2 of 2</limits.h></pre>	6-30
•	<locale.h></locale.h>	6-31
•	<pre><lwpsynch.h>*</lwpsynch.h></pre>	6-32
	<machlock.h>*</machlock.h>	6-32
Figure 6-39:		6-32
-	<sys mman.h=""></sys>	6-33
	<sys mod.h="">*</sys>	6-34
	<sys mount.h=""></sys>	6-35
-	<sys msg.h=""></sys>	6-36
	<pre><netconfig.h>, Part 1 of 2</netconfig.h></pre>	6-37
•	<pre><netconfig.h>, Part 2 of 2</netconfig.h></pre>	6-38
-	<netdir.h>, Part 1 of 2</netdir.h>	6-39
•	<netdir.h>, Part 2 of 2</netdir.h>	6-40
•	<nl_types.h></nl_types.h>	6-40
_	<sys param.h=""></sys>	6-41
Figure 6-50:		6-42
	<sys priocntl.h="">*</sys>	6-43
_	<sys procset.h=""></sys>	6-44
Figure 6-53:		6-45
	<regex.h>*, Part 1 of 2</regex.h>	6-46
	<regex.h>*, Part 2 of 2</regex.h>	6-47
_	<pre><sys resource.h=""></sys></pre>	6-48
	<pre><rpc.h>, Part 1 of 16</rpc.h></pre>	6-49
	<pre><rpc.h>, Part 2 of 16</rpc.h></pre>	6-50
•	<pre><rpc.h>, Part 3 of 16</rpc.h></pre>	6-51
	<pre><rpc.h>, Part 4 of 16</rpc.h></pre>	6-52
	<pre><rpc.h>, Part 5 of 16</rpc.h></pre>	6-53
•	<pre><rpc.h>, Part 6 of 16</rpc.h></pre>	6-54
	<pre><rpc.h>, Part 7 of 16</rpc.h></pre>	6-55
	<pre><rpc.h>, Part 8 of 16</rpc.h></pre>	6-56
rigure 6-65:	<pre><rpc.h>, Part 9 of 16</rpc.h></pre>	6-57

Table of Contents v

Figure 6-66: <pre><pre><pre><pre>Figure 6-66: <pre><pre><pre><pre><pre><pre><pre><pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>	6-58
Figure 6-67: <rpc.h>, Part 11 of 16</rpc.h>	6-59
Figure 6-68: <rpc.h>, Part 12 of 16</rpc.h>	6-60
Figure 6-69: <rpc.h>, Part 13 of 16</rpc.h>	6-60
Figure 6-70: <rpc.h>, Part 14 of 16</rpc.h>	6-61
Figure 6-71: <rpc.h>, Part 15 of 16</rpc.h>	6-62
Figure 6-72: <pre><pre>rpc.h></pre>, Part 16 of 16</pre>	6-63
Figure 6-73: <rtpriocntl.h>*</rtpriocntl.h>	6-63
Figure 6-74: <search.h></search.h>	6-64
Figure 6-75: <sys sem.h=""></sys>	6-65
Figure 6-76: <setjmp.h></setjmp.h>	6-66
Figure 6-77: <sys shm.h=""></sys>	6-66
Figure 6-78: <signal.h>, Part 1 of 3</signal.h>	6-67
Figure 6-79: <signal.h>, Part 2 of 3</signal.h>	6-68
Figure 6-80: <signal.h>, Part 3 of 3</signal.h>	6-69
Figure 6-81: <sys siginfo.h="">, Part 1 of 5</sys>	6-69
Figure 6-82: <sys siginfo.h="">, Part 2 of 5</sys>	6-70
Figure 6-83: <sys siginfo.h="">, Part 3 of 5</sys>	6-71
Figure 6-84: <sys siginfo.h="">, Part 4 of 5</sys>	6-72
Figure 6-85: <sys siginfo.h="">*, Part 5 of 5</sys>	6-72
Figure 6-86: <sys stat.h="">, Part 1 of 2</sys>	6-73
Figure 6-87: <sys stat.h="">, Part 2 of 2</sys>	6-74
Figure 6-88: <sys statvfs.h=""></sys>	6-75
Figure 6-89: <stdarg.h></stdarg.h>	6-76
Figure 6-90: <stddef.h></stddef.h>	6-76
Figure 6-91: <stdio.h>, Part 0 of 2</stdio.h>	6-77
Figure 6-92: <stdio.h>, Part 2 of 2</stdio.h>	6-78
Figure 6-93: <stdlib.h></stdlib.h>	6-79 6-80
Figure 6-94: <stropts.h>, Part 1 of 6 Figure 6-95: <stropts.h>, Part 2 of 6</stropts.h></stropts.h>	6-81
	6-82
Figure 6-96: <stropts.h>, Part 3 of 6 Figure 6-97: <stropts.h>, Part 4 of 6</stropts.h></stropts.h>	6-83
Figure 6-98: <stropts.h>, Part 5 of 6</stropts.h>	6-84
Figure 6-99: <stropts.h>, Part 6 of 6</stropts.h>	6-85
Figure 6-100: <synch.h>*, Part 1 of 3</synch.h>	6-86
Figure 6-101: <synch.h>*, Part 2 of 3</synch.h>	6-87
Figure 6-102: <synch.h>*, Part 3 of 3</synch.h>	6-88
Figure 6-103: <sys sysi86.h=""></sys>	6-88
Figure 6-104: <termios.h>, Part 1 of 10</termios.h>	6-89
Figure 6-105: <termios.h>, Part 2 of 10</termios.h>	6-90
Figure 6-106: <termios.h>, Part 3 of 10</termios.h>	6-91
Figure 6-107: <termios.h>, Part 4 of 10</termios.h>	6-92
Figure 6-108: <termios.h>, Part 5 of 10</termios.h>	6-93
Figure 6-109: <termios.h>, Part 6 of 10</termios.h>	6-94
Figure 6-110: <termios.h>, Part 7 of 10</termios.h>	6-95
-	

vi Table of Contents

Figure 6-111:	<termios.h>, Part 8 of 10</termios.h>	6-96
Figure 6-112:	<termios.h>, Part 9 of 10</termios.h>	6-97
Figure 6-113:	<termios.h>, Part 10 of 10</termios.h>	6-98
Figure 6-114:	<thread.h>*, Part 1 of 2</thread.h>	6-99
Figure 6-115:	<thread.h>*, Part 2 of 2</thread.h>	6-100
Figure 6-116:	<sys ticlts.h=""></sys>	6-100
Figure 6-117:	<sys ticots.h=""></sys>	6-101
Figure 6-118:	<sys ticotsord.h=""></sys>	6-101
Figure 6-119:		6-102
Figure 6-120:	<sys time.h=""></sys>	6-103
Figure 6-121:	<sys times.h=""></sys>	6-104
	<tiuser.h>, Error Return Values</tiuser.h>	6-105
	<tiuser.h>, Event Bitmasks</tiuser.h>	6-106
	<tiuser.h>, Flags</tiuser.h>	6-106
	<tiuser.h>, Service Types</tiuser.h>	6-107
Figure 6-126:	<pre><tiuser.h>, Values for flags field in t_info structure</tiuser.h></pre>	6-107
•	<pre><tiuser.h>, Transport Interface Data Structures, 1 of 2</tiuser.h></pre>	6-108
	<pre><tiuser.h>, Transport Interface Data Structures, 2 of 2</tiuser.h></pre>	6-109
	<tiuser.h>, Structure Types</tiuser.h>	6-110
•	<pre><tiuser.h>, Fields of Structures</tiuser.h></pre>	6-110
	<tiuser.h>, Transport Interface States</tiuser.h>	6-111
•	<pre><tiuser.h>, User-level Events</tiuser.h></pre>	6-112
Figure 6-133:	<tspriocntl.h>*</tspriocntl.h>	6-113
	<sys types.h=""></sys>	6-114
_	<ucontext.h>, Part 1 of 2</ucontext.h>	6-115
Figure 6-136:	<ucontext.h>, Part 2 of 2</ucontext.h>	6-116
•	<sys uio.h=""></sys>	6-117
Figure 6-138:		6-117
•	<unistd.h>, Part 1 of 2</unistd.h>	6-117
	<unistd.h>, Part 2 of 2</unistd.h>	6-119
Figure 6-141:		6-119
_	<sys utsname.h=""></sys>	6-120
Figure 6-143:		6-121
Figure 6-144:		6-122
	<wctype.h>*, Part 1 of 3</wctype.h>	6-123
	<pre><wctype.h>*, Part 2 of 3</wctype.h></pre>	6-124
	<wctype.h>*, Part 3 of 3</wctype.h>	6-125
•	<pre><wordexp.h>*</wordexp.h></pre>	6-126
_	<x11 atom.h="">, Part 1 of 3</x11>	6-128
•	<x11 atom.h="">, Part 2 of 3</x11>	6-129
_	<x11 atom.h="">, Part 3 of 3</x11>	6-130
J	<x11 composite.h=""></x11>	6-131
_	<x11 constraint.h=""></x11>	6-131
-	<x11 core.h=""></x11>	6-131
Figure 6-155:	<x11 cursorfont.h="">, Part 1 of 3</x11>	6-132

Table of Contents vii

Figure 6-156:	<x11 cursorfont.h="">, Part 2 of 3</x11>	6-133
•	<x11 cursorfont.h="">, Part 3 of 3</x11>	6-134
•	<x11 cursorione.in="">, Part 1 of 6</x11>	6-135
•	<x11 intrinsic.h="">, Part 2 of 6</x11>	6-136
•	<x11 intrinsic.h="">, Part 3 of 6</x11>	6-137
•	<x11 intrinsic.h="">, Part 4 of 6</x11>	6-138
•	<x11 intrinsic.h="">, Part 5 of 6</x11>	6-139
•	<x11 intrinsic.h="">, Part 6 of 6</x11>	6-140
	<x11 object.h=""></x11>	6-140
_	<x11 rectobj.h=""></x11>	6-140
	<x11 extensions="" shape.h="">*</x11>	6-141
	<x11 shell.h=""></x11>	6-141
•	<x11 vendor.h=""></x11>	6-141
	<x11 x.h="">, Part 1 of 12</x11>	6-142
•	<x11 x.h="">, Part 2 of 12</x11>	6-143
_	<x11 x.h="">, Part 3 of 12</x11>	6-144
Figure 6-172:	<x11 x.h="">, Part 4 of 12</x11>	6-145
Figure 6-173:	<x11 x.h="">, Part 5 of 12</x11>	6-146
Figure 6-174:	<x11 x.h="">, Part 6 of 12</x11>	6-147
Figure 6-175:	<x11 x.h="">, Part 7 of 12</x11>	6-148
Figure 6-176:	<x11 x.h="">, Part 8 of 12</x11>	6-149
Figure 6-177:	<x11 x.h="">, Part 9 of 12</x11>	6-150
Figure 6-178:	<x11 x.h="">, Part 10 of 12</x11>	6-151
Figure 6-179:	<x11 x.h="">, Part 11 of 12</x11>	6-152
Figure 6-180:	<x11 x.h="">, Part 12 of 12</x11>	6-153
Figure 6-181:	<x11 xcms.h="">, Part 1 of 5</x11>	6-154
•	<x11 xcms.h="">, Part 2 of 5</x11>	6-155
Figure 6-183:	<x11 xcms.h="">, Part 3 of 5</x11>	6-156
•	<x11 xcms.h="">, Part 4 of 5</x11>	6-157
•	<x11 xcms.h="">, Part 5 of 5</x11>	6-158
Figure 6-186:	<x11 xlib.h="">, Part 1 of 27</x11>	6-159
•	<x11 xlib.h="">, Part 2 of 27</x11>	6-159
•	<x11 xlib.h="">, Part 3 of 27</x11>	6-160
-	<x11 xlib.h="">, Part 4 of 27</x11>	6-161
•	<x11 xlib.h="">, Part 5 of 27</x11>	6-162
_	<x11 xlib.h="">, Part 6 of 27</x11>	6-163
•	<x11 xlib.h="">, Part 7 of 27</x11>	6-164
•	<x11 xlib.h="">, Part 8 of 27</x11>	6-165
•	<x11 xlib.h="">, Part 9 of 27</x11>	6-166
•	<x11 xlib.h="">, Part 10 of 27</x11>	6-167
_	<x11 xlib.h="">, Part 11 of 27</x11>	6-168
•	<x11 xlib.h="">, Part 12 of 27</x11>	6-169
	<x11 xlib.h="">, Part 13 of 27</x11>	6-170
_	<x11 xlib.h="">, Part 14 of 27</x11>	6-171
Figure 6-200:	<x11 xlib.h="">, Part 15 of 27</x11>	6-172

viii Table of Contents

F : 0.004	P (40, 107	0.470
_	<x11 x1ib.h="">, Part 16 of 27</x11>	6-173
_	<x11 xlib.h="">, Part 17 of 27</x11>	6-174
_	<x11 xlib.h="">, Part 18 of 27</x11>	6-175
•	<x11 xlib.h="">, Part 19 of 27</x11>	6-176
	<x11 xlib.h="">, Part 20 of 27</x11>	6-177
-	<x11 xlib.h="">, Part 21 of 27</x11>	6-178
U	<x11 xlib.h="">, Part 22 of 27</x11>	6-179
•	<x11 xlib.h="">, Part 23 of 27</x11>	6-180
	<x11 xlib.h="">, Part 24 of 27</x11>	6-181
	<x11 xlib.h="">, Part 25 of 27</x11>	6-182
Figure 6-211:	<x11 xlib.h="">, Part 26 of 27</x11>	6-183
Figure 6-212:	<x11 xlib.h="">, Part 27 of 27</x11>	6-184
Figure 6-213:	<x11 xresource.h="">, Part 1 of 2</x11>	6-185
Figure 6-214:	<x11 xresource.h="">, Part 2 of 2</x11>	6-186
Figure 6-215:	<x11 xutil.h="">, Part 1 of 5</x11>	6-187
Figure 6-216:	<x11 xutil.h="">, Part 2 of 5</x11>	6-188
_	<x11 xutil.h="">, Part 3 of 5</x11>	6-189
	<x11 xutil.h="">, Part 4 of 5</x11>	6-190
Figure 6-219:	<x11 xutil.h="">, Part 5 of 5</x11>	6-191
_	<xm arrowb.h="">*</xm>	6-193
•	<xm arrowbg.h="">*</xm>	6-193
	<xm bulletinb.h="">*</xm>	6-193
•	<xm cascadeb.h="">*</xm>	6-193
_	<xm cascadebg.h="">*</xm>	6-194
•	<xm command.h="">*</xm>	6-194
•	<xm cutpaste.h="">*</xm>	6-195
•	<xm dialogs.h="">*</xm>	6-195
-	<xm display.h="">*</xm>	6-196
•	<xm dragc.h="">*, Part 1 of 4</xm>	6-197
•	<xm dragc.h="">*, Part 2 of 4</xm>	6-198
•	<xm dragc.h="">*, Part 3 of 4</xm>	6-199
_	<xm dragc.h="">*, Part 4 of 4</xm>	6-200
•	<xm dragicon.h="">*</xm>	6-201
•	<m></m> <m dragovers.h="">*</m>	6-201
•	<m></m> <m drawinga.h="">*</m>	6-202
•	<xm drawnb.h="">*</xm>	6-202
_	<xm dropsmgr.h="">*, Part 1 of 2</xm>	6-203
•	<pre><xm dropsmgr.h="">*, Part 2 of 2</xm></pre>	6-204
•	<pre><xm droptrans.h="">* </xm></pre>	6-205
	<xm filesb.h="">*</xm>	6-205
_	<xm form.h="">*</xm>	6-205
•	<xm frame.h="">*</xm>	6-206
•	<xm label.h="">*</xm>	6-206
•	<xm labelg.h="">*</xm>	6-206
rigure 6-245:	<xm list.h="">*</xm>	6-207

Table of Contents ix

Figure 6-246:	<xm mainw.h="">*</xm>	6-207
	<xm menushell.h="">*</xm>	6-207
	<xm messageb.h="">*</xm>	6-208
Figure 6-249:	<pre><mrm mrmpublic.h="">*, Part 1 of 3</mrm></pre>	6-209
Figure 6-250:	<pre><mrm mrmpublic.h="">*, Part 2 of 3</mrm></pre>	6-210
_	<pre><mrm mrmpublic.h="">*, Part 3 of 3</mrm></pre>	6-211
Figure 6-252:	<xm mwmutil.h="">*, Part 1 of 3</xm>	6-212
Figure 6-253:	<xm mwmutil.h="">*, Part 2 of 3</xm>	6-213
	<xm mwmutil.h="">*, Part 3 of 3</xm>	6-214
Figure 6-255:	<mm panedw.h="">*</mm>	6-214
Figure 6-256:	<xm pushb.h="">*</xm>	6-215
Figure 6-257:	<xm pushbg.h="">*</xm>	6-215
Figure 6-258:	<xm reptype.h="">*</xm>	6-215
Figure 6-259:	<xm rowcolumn.h="">*</xm>	6-216
Figure 6-260:	<xm scale.h="">*</xm>	6-216
Figure 6-261:	<xm screen.h="">*</xm>	6-216
Figure 6-262:	<xm scrollbar.h="">*</xm>	6-216
Figure 6-263:	<xm scrolledw.h="">*</xm>	6-217
Figure 6-264:	<xm selectiob.h="">*</xm>	6-217
Figure 6-265:	<xm separatog.h="">*</xm>	6-217
Figure 6-266:	<xm separator.h="">*</xm>	6-217
_	<xm text.h="">*</xm>	6-218
Figure 6-268:	<xm textf.h="">*</xm>	6-218
Figure 6-269:	<xm toggleb.h="">*</xm>	6-218
Figure 6-270:	<xm togglebg.h="">*</xm>	6-218
Figure 6-271:	<xm vendors.h="">*</xm>	6-219
•	<xm virtkeys.h="">*, Part 1 of 2</xm>	6-219
•	<xm virtkeys.h="">*, Part 2 of 2</xm>	6-220
•	<xm xm.h="">*, Part 1 of 14</xm>	6-221
•	<xm xm.h="">*, Part 2 of 14</xm>	6-222
	<xm xm.h="">*, Part 3 of 14</xm>	6-223
-	<xm xm.h="">*, Part 4 of 14</xm>	6-224
•	<xm xm.h="">*, Part 5 of 15</xm>	6-225
	<m></m> <m xm.h="">*, Part 6 of 14</m>	6-226
	<xm xm.h="">*, Part 7 of 14</xm>	6-227
_	<xm xm.h="">*, Part 8 of 14</xm>	6-228
•	<m></m> <m></m> <m></m> <m></m> <m></m> <m></m> <m></m> <m></m>	6-229
•	<m></m> <m h="" m.="">*, Part 10 of 14</m>	6-230
•	<m></m> <m xm.h="">*, Part 11 of 14</m>	6-231
	<xm xm.h="">*, Part 12 of 14</xm>	6-232
•	<m></m> <m .="" h="" xm="">*, Part 13 of 14</m>	6-233
	<xm xm.h="">*, Part 14 of 14 Xm (Ym/Xm.h) *, Part 1 of 24</xm>	6-234
	<pre><xm xmstrdefs.h="">*, Part 1 of 34</xm></pre>	6-235
_	<pre><xm xmstrdefs.h="">*, Part 2 of 34</xm></pre>	6-236
rigure 6-290:	<xm xmstrdefs.h="">*, Part 3 of 34</xm>	6-237

x Table of Contents

Figure 6-291:	<mm xmstrdefs.h="">*, Part 4 of 34</mm>	6-238
Figure 6-292:	<pre><xm xmstrdefs.h="">*, Part 5 of 34</xm></pre>	6-239
Figure 6-293:	<pre><xm xmstrdefs.h="">*, Part 6 of 34</xm></pre>	6-240
Figure 6-294:	<xm xmstrdefs.h="">*, Part 7 of 34</xm>	6-241
Figure 6-295:	<xm xmstrdefs.h="">*, Part 8 of 34</xm>	6-242
Figure 6-296:	<xm xmstrdefs.h="">*, Part 9 of 34</xm>	6-243
Figure 6-297:	<xm xmstrdefs.h="">*, Part 10 of 34</xm>	6-244
Figure 6-298:	<xm xmstrdefs.h="">*, Part 11 of 34</xm>	6-245
Figure 6-299:	<xm xmstrdefs.h="">*, Part 12 of 34</xm>	6-246
Figure 6-300:	<xm xmstrdefs.h="">*, Part 13 of 34</xm>	6-247
Figure 6-301:	<xm xmstrdefs.h="">*, Part 14 of 34</xm>	6-248
Figure 6-302:	<xm xmstrdefs.h="">*, Part 15 of 34</xm>	6-249
•	<xm xmstrdefs.h="">*, Part 16 of 34</xm>	6-250
Figure 6-304:	<xm xmstrdefs.h="">*, Part 17 of 34</xm>	6-251
•	<xm xmstrdefs.h="">*, Part 18 of 34</xm>	6-252
_	<xm xmstrdefs.h="">*, Part 19 of 34</xm>	6-253
_	<xm xmstrdefs.h="">*, Part 20 of 34</xm>	6-254
_	<xm xmstrdefs.h="">*, Part 21 of 34</xm>	6-255
•	<xm xmstrdefs.h="">*, Part 22 of 34</xm>	6-256
•	<xm xmstrdefs.h="">*, Part 23 of 34</xm>	6-257
•	<xm xmstrdefs.h="">*, Part 24 of 34</xm>	6-258
-	<xm xmstrdefs.h="">*, Part 25 of 34</xm>	6-259
•	<xm xmstrdefs.h="">*, Part 26 of 34</xm>	6-260
J	<xm xmstrdefs.h="">*, Part 27 of 34</xm>	6-261
•	<xm xmstrdefs.h="">*, Part 28 of 34</xm>	6-262
•	<xm xmstrdefs.h="">*, Part 29 of 34</xm>	6-263
_	<xm xmstrdefs.h="">*, Part 30 of 34</xm>	6-264
•	<xm xmstrdefs.h="">*, Part 31 of 34</xm>	6-265
_	<xm xmstrdefs.h="">*, Part 32 of 34</xm>	6-266
U	<xm xmstrdefs.h="">*, Part 33 of 34</xm>	6-267
•	<xm xmstrdefs.h="">*, Part 34 of 34</xm>	6-268
•	<netinet in.h=""></netinet>	6-270
•	<netinet ip.h=""></netinet>	6-271
Figure 6-324:	<netinet tcp.h=""></netinet>	6-271

Table of Contents xi

1 INTRODUCTION

The Intel296 Architecture and the Syst	om V
The Intel386 Architecture and the Syst ABI	1-1
How to Use the Intel386 Architecture A	ABI
Supplement	1-2
Evolution of the ABI Specification	1-2

Table of Contents i

DRAFT COPY March 19, 1997 File: abi_386/Cchap1 (Delta 44.3) 386:adm.book:sum

The Intel386 Architecture and the System V ABI

The *System V Application Binary Interface*, or ABI, defines a system interface for compiled application programs. Its purpose is to establish a standard binary interface for application programs on systems that implement the interfaces defined in the *System V Interface Definition, Edition 4*. This includes systems that have implemented UnixWare® 2.0.

M

This document is a supplement to the generic *System V ABI*, and it contains information specific to System V implementations built on the Intel386 processor architecture. Together, these two specifications, the generic *System V ABI* and the *Intel386 Architecture System V ABI Supplement* (hereafter referred to as the *Intel386 ABI*), constitute a complete *System V Application Binary Interface* specification for systems that implement the processor architecture of the Intel386 microprocessors.

Note that, because the Intel486 and Pentium processor are compatible members of the Intel386 architecture, this *Intel386 ABI* also applies to any system built with the Intel486 or the Pentium processor chips.

M

The Intel386 Architecture and the System V ABI

How to Use the Intel386 Architecture ABI Supplement

This document is a supplement to the generic *System V ABI* and contains information referenced in the generic specification that may differ when System V is implemented on different processors. Therefore, the generic ABI is the prime reference document, and this supplement is provided to fill gaps in that specification.

As with the *System V ABI*, this specification references other publicly-available reference documents, especially the Intel *80386 Programmer's Reference Manual*. All the information referenced by this supplement should be considered part of this specification, and just as binding as the requirements and data explicitly included here.

Evolution of the ABI Specification

The *System V Application Binary Interface* will evolve over time to address new technology and market requirements, and will be reissued at intervals of approximately three years. Each new edition of the specification is likely to contain extensions and additions that will increase the potential capabilities of applications that are written to conform to the ABI.

As with the *System V Interface Definition*, the ABI will implement **Level 1** and **Level 2** support for its constituent parts. **Level 1** support indicates that a portion of the specification will continue to be supported indefinitely, while **Level 2** support means that a portion of the specification may be withdrawn or altered after the next edition of the ABI is made available. That is, a portion of the specification moved to **Level 2** support in an edition of the ABI specification will remain in effect at least until the following edition of the specification is published.

These **Level 1** and **Level 2** classifications and qualifications apply to this Supplement, as well as to the generic specification. All components of the ABI and of this supplement have **Level 1** support unless they are explicitly labelled as **Level 2**.

The following documents may be of interest to the reader of this specification:

■ *i486 MICROPROCESSOR Programmer's Reference Manual* (Intel Literature order number 240486)

1-2 INTRODUCTION

- 80386 Programmer's Reference Manual (Intel Literature order number 230985)
- 80387 Programmer's Reference Manual (Intel Literature order number 231917)
- UnixWare® 2.0 Command Reference (a-l)
- UnixWare® 2.0 Command Reference (m-z)
- UnixWare® 2.0 Operating System API Reference: System Calls
- UnixWare® 2.0 Operating System API Reference: Library Functions
- UnixWare® 2.0 System Administration: Volumes I and II
- System V Interface Definition, Edition 4

NOTE

Diffmarkings have been retained in the text of this book to indicate in which revisions of System V certain modifications were made to the *ABI*.

A "G" character in the right hand margin indicates a change in the ABI made in UNIX System V Release 4.2.

A "M" character in the right hand margin indicates a change in the ABI made in UnixWare® 2.0.

How to Use the Intel386 Architecture ABI Supplement

2 SOFTWARE INSTALLATION

Software Distribution Formats	2-1
Physical Distribution Media	2-1
File System Formats	2-1
■ s5 File System	2-1
■ UFS File System	2-3

Table of Contents

DRAFT COPY March 19, 1997 File: abi_386/Cchap2 (Delta 44.3) 386:adm.book:sum

Software Distribution Formats

Physical Distribution Media

Approved media for physical distribution of ABI-conforming software are listed below. Inclusion of a particular medium on this list does not require an ABI-conforming system to accept that medium. For example, a conforming system may install all software through its network connection and accept none of the listed media.

- 1.44MB 3 1/2" floppy disk: quad-density, double-sided, 80 tracks/side, 18 sectors/track, 512 bytes/sector.
- 1.2MB 5 1/4" floppy disk: quad-density, double-sided, 80 tracks/side, 15 sectors/track, 512 bytes/sector.
- 360KB 5 1/4" floppy disk: double-density, double-sided, 40 tracks/side, 9 sectors/track, 512 bytes/sector.
- 60 MB quarter-inch cartridge tape in QIC-24 format.

G

■ CD-ROM optical disks.

G

■ 150 MB quarter-inch tape.



The use of 360KB 5 1/4" floppy disk, and 60 MB quarter inch cartridge tape as media for application distribution is moved to Level 2 as of January 1, 1993.

File System Formats

Every file system storage volume must conform to a supported format. Two formats are supported: s5 and ufs.

s5 File System

The first physical block on the medium should be empty, and the second contains the device's *superblock*. The third contains an inode list, and remaining blocks on the device contain data. The *superblock* has the following format:

Software Distribution Formats

```
#define NICFREE
                          50
#define NICINOD
                          100
struct filsys {
        u_short s_isize;
        daddr_t s_fsize;
        short
                         s_nfree;
        daddr_t s_free[NICFREE];
        short
                        s ninode;
        ushort_t s_inode[NICINOD];
                       s_flock;
        char
        char
                          s_ilock;
        char
                        s_fmod;
        char
                        s_ronly;
        time_t
                         s_time;
                         s_dinfo[4];
        short.
        daddr_t s_tfree;
        ushort_t s_tinode;
        char
                         s_fname[6];
                         s_fpack[6];
                        s_fill[12];
        long
        long
                         s_state;
        long
                         s magic;
        long
                         s_type;
#define FsMAGIC
                 0xfd187e20
#define Fslb
                          1
#define Fs2b
                          2
#define Fs4b
                          3
                 0x7c269d38
#define FsOKAY
#define FsACTIVE 0x5e72d81a
                          0xcb096f43
#define FsBAD
#define FsBADBLK 0xbadbc14b
```

- *s_type* indicates the file system type. Currently, three types of file systems are supported: the original 512-byte logical block, the 1024-byte logical block, and the 2048-byte logical block. *s_magic* is used to distinguish the original 512-byte oriented file systems from the newer file systems. If this field is not equal to the magic number, fsmagic, the type is assumed to be fslb, otherwise the *s_type* field is used.
- s_state indicates the state of the file system. A cleanly unmounted, undamaged file system is indicated by the FSOKAY state. After a file system has been mounted for update, the state changes to FSACTIVE.

2-2

SOFTWARE INSTALLATION

- *s_isize* is the address of the first data block after the i-list; the i-list starts just after the super-block, namely in block 2; thus the i-list is *s_isize-2* blocks long.
- s fsize is the first block not potentially available for allocation to a file.

The free list for each volume is maintained as follows. The *s_free* array contains up to 49 numbers of free blocks. *s_free*[0] is the block number of the head of a chain of blocks constituting the free list. The first long in each free-chain block is the number (up to 50) of free-block numbers listed in the next 50 longs of this chain member. The first of these 50 blocks is the link to the next member of the chain.

- *s_tfree* is the total free blocks available in the file system.
- *s ninode* is the number of free i-numbers in the *s inode* array.
- *s tinode* is the total free i-nodes available in the file system.
- *s_flock* and *s_ilock* are flags maintained in the core copy of the file system. *s_fmod* is a flag that indicates that the super-block has changed and should be copied to the disk during the next periodic update of file system information.
- *s_ronly* is a read-only flag to indicate write-protection.
- *s_time* is the last time the super-block of the file system was changed, and is the number of seconds that have elapsed since 00:00 Jan. 1, 1970 (GMT).
- *s fname* is the name of the file system and *s fpack* is the name of the pack.

I-numbers begin at 1, and the storage for i-nodes begins in block 2. I-node 1 is reserved for future use. I-node 2 is reserved for the root directory of the file system, but no other i-number has a built-in meaning. Each i-node represents one file.

UFS File System

In the UFS file system, the first physical block on the device should be empty, and the second contains the *superblock* for the file system. Remaining blocks contain data

The ufs *superblock* contains an *fs* data structure. This structure, and other relevant data objects are defined below.

Software Distribution Formats

```
struct csum {
                        cs_ndir;
         long
         long
                        cs_nbfree;
                        cs_nifree;
         long
         long
                        cs_nffree;
};
struct fs {
                        *fs_link;
         struct fs
         struct fs
                        *fs_rlink;
         daddr_t
                        fs_sblkno;
                        fs_cblkno;
         daddr_t
         daddr_t
                        fs_iblkno;
         daddr_t
                        fs_dblkno;
         long
                        fs_cgoffset;
         long
                        fs_cgmask;
         time_t
                        fs_time;
         long
                        fs_size;
         long
                        fs_dsize;
         long
                        fs_ncg;
                        fs_bsize;
         long
         long
                        fs_fsize;
         long
                        fs_frag;
         long
                        fs_minfree;
         long
                        fs_rotdelay;
                        fs_rps;
         long
         long
                        fs_bmask;
                        fs_fmask;
         long
                        fs_bshift;
         long
         long
                        fs_fshift;
         long
                        fs_maxcontig;
         long
                        fs_maxbpg;
                        fs_fragshift;
         long
         long
                        fs_fsbtodb;
         long
                        fs_sbsize;
         long
                        fs_csmask;
         long
                        fs_csshift;
         long
                        fs_nindir;
         long
                        fs_inopb;
         long
                        fs_nspf;
         long
                        fs_optim;
         long
                        fs_state;
                        fs_sparecon[2];
         long
         long
                        fs_id[2];
         daddr_t
                        fs_csaddr;
                        fs_cssize;
         long
         long
                        fs_cgsize;
         long
                        fs_ntrak;
          long
                        fs_nsect;
                        fs_spc;
          long
          long
                        fs_ncyl;
```

(continued on next page)

SOFTWARE INSTALLATION

```
long
                        fs_cpg;
         long
                        fs_ipg;
         long
                        fs_fpg;
         struct csum
                       fs_cstotal;
         char
                        fs_fmod;
                       fs_clean;
         char
         char
                       fs_ronly;
         char
                       fs_flags;
         char
                       fs_fsmnt[MAXMNTLEN];
         long
                        fs_cgrotor;
                       *fs_csp[MAXCSBUFS];
         struct csum
         long
                        fs_cpc;
                       fs_postbl[MAXCPG][NRPOS];
         short
                       fs_magic;
         long
         u_char
                        fs_rotbl[1];
};
struct cg {
         struct cg
                        *cg_link;
         struct cg
                        *cg_rlink;
                       cg_time;
         time_t
         long
                        cg_cgx;
         short
                       cg_ncyl;
         short
                        cg_niblk;
         long
                       cg_ndblk;
         struct csum
                      cg_cs;
         long
                       cg_rotor;
                       cg_frotor;
         long
         long
                       cg_irotor;
                       cg_frsum[MAXFRAG];
         long
         long
                        cg_btot[MAXCPG];
         short
                        cg_b[MAXCPG][NRPOS];
         char
                       cg_iused[MAXIPG/NBBY];
         long
                        cg_magic;
         u_char
                        cg_free[1];
};
#define FS_MAGIC
                        0x011954
#define BBSIZE
                        8192
#define SBSIZE
                        8192
#define BBLOCK
                        ((daddr_t)(0))
#define SBLOCK
                        ((daddr_t)(BBLOCK + BBSIZE / DEV_BSIZE))
#define UFSROOTINO
                        ((ino t)2)
#define LOSTFOUNDINO
                        (UFSROOTINO + 1)
#define NRPOS
                        8
                        2048
#define MAXIPG
#define MINBSIZE
                        4096
#define MAXCPG
                        32
#define MAXMNTLEN
                        512
#define MAXCSBUFS
                        32
#define FS_OPTTIME
                        0
```

(continued on next page)

Software Distribution Formats

#define FS_OPTSPACE 1
#define MAXBPC (SBSIZE - sizeof (struct fs))
#define CG_MAGIC 0x090255

CAUTION

The distribution of software in filesystem format is Level 2 as of January 1, 1993

SOFTWARE INSTALLATION

3 LOW-LEVEL SYSTEM INFORMATION

Machine Interface Processor Architecture Data Representation ■ Fundamental Types ■ Aggregates and Unions ■ Bit-Fields	3-1 3-1 3-1 3-2 3-3 3-6
Function Calling Sequence	3-9
Registers and the Stack Frame	3-9
Functions Returning Scalars or No Value	3-12
Functions Returning Structures or Unions	3-14
Integral and Pointer Arguments	3-17
Floating-Point Arguments	3-17
Structure and Union Arguments	3-18
Operating System Interface	3-20
Virtual Address Space	3-20
■ Page Size	3-20
■ Virtual Address Assignments	3-20
Managing the Process Stack	3-22
■ Coding Guidelines	3-23
Processor Execution Modes	3-24
Exception Interface	3-24
■ Hardware Exception Types	3-24
■ Software Trap Types	3-25
Process Initialization ■ Special Registers	3-26 3-26
■ Process Stack and Registers	3-20 3-28
=	5 <u>2</u> 0

Table of Contents i

DRAFT COPY March 19, 1997 File: abi_386/Cchap3 (Delta 44.4) 386:adm.book:sum

Coding Examples	3-34
Code Model Overview	3-35
Position-Independent Function Prologue	3-36
Data Objects	3-37
Function Calls	3-39
Branching	3-41
C Stack Frame	3-43
Variable Argument List	3-44
Allocating Stack Space Dynamically	3-44

ii Table of Contents

DRAFT COPY March 19, 1997 File: abi_386/Cchap3 (Delta 44.4) 386:adm.book:sum

Machine Interface

Processor Architecture

The Intel 80386 Programmer's Reference Manual (Intel Literature order number 230985) and the Intel 80387 Programmer's Reference Manual (Intel Literature order number 231917) together define the processor architecture. The architecture of the combined Intel386/Intel 387 processors is hereafter referred to as the Intel386 architecture. Programs intended to execute directly on the processor use the instruction set, instruction encodings, and instruction semantics of the architecture. Three points deserve explicit mention.

- A program may assume all documented instructions exist.
- A program may assume all documented instructions work.
- A program may use only the instructions defined by the architecture.

In other words, *from a program's perspective*, the execution environment provides a complete and working implementation of the Intel386 architecture.

This does not imply that the underlying implementation provides all instructions in hardware, only that the instructions perform the specified operations and produce the specified results. The ABI neither places performance constraints on systems nor specifies what instructions must be implemented in hardware. A software emulation of the architecture could conform to the ABI.

Some processors might support the Intel386 architecture as a subset, providing additional instructions or capabilities. Programs that use those capabilities explicitly do not conform to the Intel386 ABI. Executing those programs on machines without the additional capabilities gives undefined behavior.

Data Representation

Within this specification, the term *halfword* refers to a 16-bit object, the term *word* refers to a 32-bit object, and the term *doubleword* refers to a 64-bit object.

Machine Interface 3-1

Fundamental Types

Figure 3-1 shows the correspondence between ANSI C's scalar types and the processor's.

Figure 3-1: Scalar Types

			Alignment	Intel386
Type	C	sizeof	(bytes)	Architecture
	char signed char	1	1	signed byte
	unsigned char	1	1	unsigned byte
	short signed short	2	2	signed halfword
	unsigned short	2	2	unsigned halfword
Integral	int signed int long signed long enum	4	4	signed word
	unsigned int unsigned long	4	4	unsigned word
Pointer	any-type * any-type (*)()	4	4	unsigned word
	float	4	4	single-precision (IEEE)
Floating-point	double	8	4	double-precision (IEEE)
	long double	12	4	extended-precision (IEEE)

NOTE

The Intel386 architecture does not require doubleword alignment for double-precision values. Nevertheless, for data structure compatibility with other Intel architectures, compilers may provide a method to align double-precision values on doubleword boundaries.

LOW-LEVEL SYSTEM INFORMATION



CAUTION A compiler that provides the doubleword alignment mentioned above can generate code (data structures and function calling sequences) that do not conform to the Intel386 ABI. Programs built with the doubleword alignment facility can thus violate conformance to the Intel386 ABI. See "Aggregates and Unions" below and "Function Calling Sequence" later in this chapter for more information.

A null pointer (for all types) has the value zero.

The Intel386 architecture does not require all data access to be properly aligned. For example, double-precision values occupy 1 doubleword (8-bytes), and their natural alignment is a word boundary, meaning their addresses are multiples of 4. Compilers should allocate independent data objects with the proper alignment; examples include global arrays of double-precision variables, FORTRAN COMMON blocks, and unconstrained stack objects. However, some language facilities (such as FORTRAN EQUIVALENCE statements) may create objects with only byte alignment. Consequently, arbitrary data accesses, such as pointers dereference or reference arguments, might or might not be properly aligned. Accessing misaligned data will be slower than accessing properly aligned data, but otherwise there is no difference.

Aggregates and Unions

Aggregates (structures and arrays) and unions assume the alignment of their most strictly aligned component. The size of any object, including aggregates and unions, is always a multiple of the object's alignment. An array uses the same alignment as its elements. Structure and union objects can require padding to meet size and alignment constraints. The contents of any padding is undefined.

G

- An entire structure or union object is aligned on the same boundary as its most strictly aligned member.
- Each member is assigned to the lowest available offset with the appropriate alignment. This may require internal padding, depending on the previous member.
- A structure's size is increased, if necessary, to make it a multiple of the alignment. This may require tail padding, depending on the last member.

Machine Interface 3-3



ABI conformant code may not read or modify anything marked reserved or padding.

Μ

In the following examples, members' byte offsets appear in the upper right corners.

Figure 3-2: Structure Smaller Than a Word

```
struct {
     char c;
};
Byte aligned, sizeof is 1

c

c

c
```

Figure 3-3: No Padding

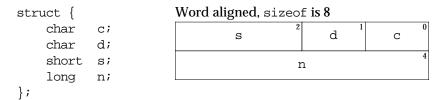
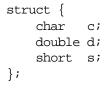


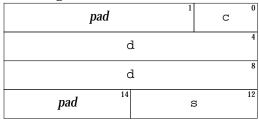
Figure 3-4: Internal Padding

LOW-LEVEL SYSTEM INFORMATION

Figure 3-5: Internal and Tail Padding



Word aligned, sizeof is 16



NOTE

The Intel386 architecture does not require doubleword alignment for doubleprecision values. Nevertheless, for data structure compatibility with other Intel architectures, compilers may provide a method to align double-precision values on doubleword boundaries.

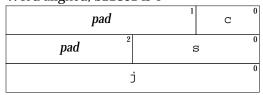


CAUTION A compiler that provides the doubleword alignment mentioned above would arrange the preceding structure differently. Programs built with the doubleword alignment facility would not conform to the Intel386 ABI, and they would not be data-compatible with conforming Intel386 programs.

Figure 3-6: union Allocation

```
union {
    char
            Сį
    short s;
    int
            j;
};
```

Word aligned, sizeof is 4



Machine Interface 3-5

Bit-Fields

C struct and union definitions may have *bit-fields*, which define integral objects with a specified number of bits.

Figure 3-7: Bit-Field Ranges

Bit-field Type	Width w	Range	
signed char		-2^{w-1} to $2^{w-1}-1$	
char	1 to 8	0 to $2^{w} - 1$	
unsigned char		0 to $2^{w} - 1$	
signed short		-2^{w-1} to $2^{w-1}-1$	
short	1 to 16	0 to $2^{w} - 1$	
unsigned short	-	0 to $2^{w} - 1$	
signed int		-2^{w-1} to $2^{w-1}-1$	
int	1 to 32	0 to $2^{w} - 1$	
enum	1 10 32	0 to $2^{w} - 1$	
unsigned int		0 to $2^{w} - 1$	
signed long		-2^{w-1} to $2^{w-1}-1$	
long	1 to 32	0 to $2^{w} - 1$	
unsigned long		0 to $2^{w} - 1$	

"Plain" bit-fields (that is, those neither signed nor unsigned) always have non-negative values. Although they may have type char, short, int, or long (which can have negative values), these bit-fields have the same range as a bit-field of the same size with the corresponding unsigned type. Bit-fields obey the same size and alignment rules as other structure and union members, with the following additions:

- Bit-fields are allocated from right to left (least to most significant).
- A bit-field must entirely reside in a storage unit appropriate for its declared type. Thus a bit-field never crosses its unit boundary.
- Bit-fields may share a storage unit with other struct/union members, including members that are not bit-fields. Of course, struct members occupy different parts of the storage unit.
- Unnamed bit-fields' types do not affect the alignment of a structure or union, although individual bit-fields' member offsets obey the alignment constraints.

LOW-LEVEL SYSTEM INFORMATION

The following examples show struct and union members' byte offsets in the upper right corners; bit numbers appear in the lower corners.

Figure 3-8: Bit Numbering



Figure 3-9: Right-to-Left Allocation

```
struct {
                         Word aligned, sizeof is 4
    int
             j:5;
                                pad
    int
            k:6;
    int
            m:7;
};
```

Figure 3-10: Boundary Alignment

struct {

int

char

short

short

char

};

```
Word aligned, sizeof is 12
short s:9;
                                pad
                                          j
                        С
                                                     s
        j:9;
        c;
                                          pad
                      pad
                                                     t
                                 u
       t:9;
       u:9;
                                 pad
                                                      d
       d;
```

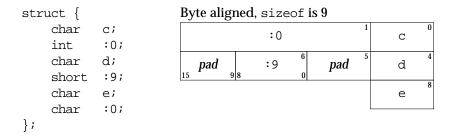
3-7 **Machine Interface**

Figure 3-11: Storage Unit Sharing

```
struct { Halfword aligned, sizeof is 2 char c; short s:8; \begin{bmatrix} & & & & & & & \\ & char & c; & & & & & \\ & s & & & c & & \\ & & & s & & c & & \\ & & & & & & c & \\ \end{bmatrix}_{15}
```

Figure 3-12: union Allocation

Figure 3-13: Unnamed Bit-Fields



As the examples show, int bit-fields (including signed and unsigned) pack more densely than smaller base types. One can use char and short bit-fields to force particular alignments, but int is generally more efficient.

LOW-LEVEL SYSTEM INFORMATION

Function Calling Sequence

This section discusses the standard function calling sequence, including stack frame layout, register usage, parameter passing, and so on. The system libraries described in Chapter 6 require this calling sequence.



The standard calling sequence *requirements* apply only to global functions. Local functions that are not reachable from other compilation units may use different conventions. Nonetheless, it is recommended that all functions use the standard calling sequence when possible.



C programs follow the conventions given here. For specific information on the implementation of C, see "Coding Examples" in this chapter.

Registers and the Stack Frame

The Intel386 architecture provides a number of registers. All the integer registers and all the floating-point registers are global to all procedures in a running program.

Brief register descriptions appear in Figure 3-14 more complete information appears later.

Function Calling Sequence

Figure 3-14: Processor Registers

Type	Name	Usage	
	%eax	Return value	
	%edx	Dividend register (divide operations)	
	%ecx	Count register (shift and string operations)	
General	%ebx	Local register variable	
General	%ebp	Stack frame pointer (optional)	
	%esi	Local register variable	
	%edi	Local register variable	
	%esp	Stack pointer	
	%st(0)	floating-point stack top, return value	
The street of	%st(1)	floating-point next to stack top	
Floating-point		•	
	%st(7)	floating-point stack bottom	

G

In addition to registers, each function has a frame on the run-time stack. This stack grows downward from high addresses. Figure 3-15 shows the stack frame organization.

Figure 3-15: Standard Stack Frame

Position	Contents Frame		
4n+8(%ebp)	argument word n		High addresses
	Previous		
8(%ebp)	argument word 0		
4(%ebp)	return address		
0(%ebp)	previous %ebp (optional)		
-4(%ebp)	-4(%ebp) unspecified Current		
0(%esp)	variable size		Low addresses

Several key points about the stack frame deserve mention.

■ The stack is word aligned. Although the architecture does not require any alignment of the stack, software convention and the operating system requires that the stack be aligned on a word boundary.

G

LOW-LEVEL SYSTEM INFORMATION

- Argument words are pushed onto the stack in reverse order (that is, the rightmost argument in C call syntax has the highest address), preserving the stack's word alignment. All incoming arguments appear on the stack, residing in the stack frame of the caller.
- An argument's size is increased, if necessary, to make it a multiple of words. This may require tail padding, depending on the size of the argument.
- Other areas depend on the compiler and the code being compiled. The standard calling sequence does not define a maximum stack frame size, nor does it restrict how a language system uses the "unspecified" area of the standard stack frame.

All registers on the Intel386 are global and thus visible to both a calling and a called function. Registers <code>%ebp</code>, <code>%ebx</code>, <code>%edi</code>, <code>%esi</code>, and <code>%esp</code> ''belong'' to the calling function. In other words, a called function must preserve these registers' values for its caller. Remaining registers ''belong'' to the called function. If a calling function wants to preserve such a register value across a function call, it must save the value in its local stack frame.

Some registers have assigned roles in the standard calling sequence:

%esp	The stack pointer holds the limit of the current stack frame, which
	is the address of the stack's bottom-most, valid word. At all
	times, the stack pointer should point to a word-aligned area.

%ebp	The <i>frame pointer</i> optionally holds a base address for the current
	stack frame. Consequently, a function has registers pointing to
	both ends of its frame. Incoming arguments reside in the previ-
	ous frame, referenced as positive offsets from %ebp, while local
	variables reside in the current frame, referenced as negative
	offsets from %ebp. A function must preserve this register's value
	for its caller.

Integral and pointer return values appear in %eax. A function that
returns a struct or union value places the address of the result
in %eax. Otherwise this is a scratch register.

As described below, this register serves as the <i>global offset table</i> base register for position-independent code. For absolute code,
%ebx serves as a local register and has no specified role in the
function calling sequence. In either case, a function must
preserve the register value for the caller

%esi and %edi	These <i>local</i>	registers have no	specified role	in the function (calling
	sequence.	A function must	preserve their	values for the c	<mark>aller.</mark>

Function Calling Sequence

%eax

%ebx

3-11

G

%ecx and %edx

Scratch registers have no specified role in the standard calling sequence. Functions do not have to preserve their values for the caller.

%st(0)

Floating-point return values appear on the top of the floating-point register stack; there is no difference in the representation of single- or double-precision values in floating-point registers. If the function does not return a floating-point value, then this register must be empty. This register must be empty before entry to a function.

G

%st(1) through %st(7)

Floating-point scratch registers have no specified role in the standard calling sequence. These registers must be empty before entry and upon exit from a function.

EFLAGS

The *flags register* contains the system flags, such as the direction flag and the carry flag. The direction flag must be set to the "forward" (that is, zero) direction before entry and upon exit from a function. Other user flags have no specified role in the standard calling sequence and are not preserved.

Floating-Point Control Word

The Intel387 *control word* contains the floating-point flags, such as the rounding mode and exception masking.

Signals can interrupt processes [see signal(BA_OS)]. Functions called during signal handling have no unusual restrictions on their use of registers. Moreover, if a signal handling function returns, the process resumes its original execution path with registers restored to their original values. Thus, programs and compilers may freely use all registers without the danger of signal handlers changing their values.

Functions Returning Scalars or No Value

A function that returns an integral or pointer value places its result in register %eax.

A floating-point return value appears on the top of the Intel387 register stack. The caller then must remove the value from the Intel387 stack, even if it doesn't use the value. Failure of either side to meet its obligations leads to undefined program behavior. The standard calling sequence does not include any method to detect such failures nor to detect return value type mismatches. Therefore the user must declare all functions properly. There is no difference in the representation of

LOW-LEVEL SYSTEM INFORMATION

single-, double- or extended-precision values in floating-point registers.

Functions that return no value (also called procedures or void functions) put no particular value in any register.

A call instruction pushes the address of the next instruction (the return address) onto the stack. The ret instruction pops the address off the stack and effectively continues execution at the next instruction after the call instruction. A function that returns a scalar or no value must preserve the caller's registers as described earlier. Additionally, the called function must remove the return address from the stack, leaving the stack pointer (%esp) with the value it had before the call instruction was executed.

To illustrate, the following function prologue allocates 80 bytes of local stack space and saves the local registers %ebx, %esi, and %edi.

Figure 3-16: Function Prologue

```
prologue:

pushl %ebp / save frame pointer
movl %esp, %ebp / set new frame pointer
subl $80, %esp / allocate stack space
pushl %edi / save local register
pushl %esi / save local register
pushl %ebx / save local register
```

An epilogue for the example that restores the state for the caller. This example returns the value in %edi by moving it to %eax.

Function Calling Sequence

Figure 3-17: Function Epilogue

```
movl
             %edi, %eax
                           / set up return value
epilogue:
             %ebx
                           / restore local register
      popl
      popl
             %esi
                           / restore local register
                           / restore local register
      popl
                           / restore frame pointer
      leave
      ret
                           / pop return address
```

NOTE

Although some functions can be optimized to eliminate the save and restore of the frame pointer, the general case uses the standard prologue and epiloque.

Sections below describe where arguments appear on the stack. The examples are written as if the function prologue described above had been used.

Position-independent code uses the *ebx register to hold the address of the global offset table. If a function needs the global offset table's address, either directly or indirectly, it is responsible for computing the value. See "Coding Examples" later in this chapter and "Dynamic Linking" in Chapter 5 for more information.

Functions Returning Structures or Unions

If a function returns a structure or union, then the caller provides space for the return value and places its address on the stack as argument word zero. In effect, this address becomes a "hidden" first argument. Having the caller supply the return object's space allows re-entrancy.



Structures and unions in this context have fixed sizes. The ABI does not specify how to handle variable sized objects.

LOW-LEVEL SYSTEM INFORMATION

A function that returns a structure or union also sets <code>%eax</code> to the value of the original address of the caller's area before it returns. Thus when the caller receives control again, the address of the returned object resides in register <code>%eax</code> and can be used to access the object. Both the calling and the called functions must cooperate to pass the return value successfully:

- The calling function must supply space for the return value and pass its address in the stack frame:
- The called function must use the address from the frame and copy the return value to the object so supplied;
- The called function must remove this address from the stack before returning.

Failure of either side to meet its obligations leads to undefined program behavior. The standard function calling sequence does not include any method to detect such failures nor to detect structure and union type mismatches. Therefore the user must declare all functions properly.

Figure 3-18 illustrates the stack contents when the function receives control (after the call instruction) and when the calling function again receives control (after the ret instruction).

Figure 3-18: Stack Contents for Functions Returning struct/union

Position	After call	After ret	Position
4 n +4(%esp)	argument word n	argument word n	4 <i>n</i> -4(%esp)
8(%esp) 4(%esp)	argument word 1 value address	argument word 1	0(%esp)
0(%esp)	return address		

To illustrate, the following function prologue allocates 80 bytes of local stack space and saves the local registers %ebx, %esi, and %edi. Additionally, it removes the "hidden" argument from the stack and saves it in the highest word of the local stack frame.

Function Calling Sequence

Figure 3-19: Function Prologue (Returning struct/union)

```
prologue:
                          / pop return address
      popl
            %eax
      xchgl %eax, 0(%esp) / swap return address
                             / and return value address
      pushl %ebp
                          / save frame pointer
            %esp, %ebp
                          / set new frame pointer
      movl
      subl $80, %esp
                         / allocate local space
      pushl %edi
                         / save local register
      pushl %esi
                         / save local register
      pushl %ebx
                          / save local register
      movl %eax, -4(%ebp) / save return value address
```

An epilogue for the example that restores the state for the caller.

Figure 3-20: Function Epilogue

```
epilogue:
      movl
            -4(%ebp), %eax
                                / set up return value
      popl
             %ebx
                                / restore local register
                                / restore local register
      popl
             %esi
      popl
             %edi
                                 / restore local register
      leave
                                / restore frame pointer
      ret
                                / pop return address
```

LOW-LEVEL SYSTEM INFORMATION



Although some functions can be optimized to eliminate the save and restore of the frame pointer, the general case uses the standard prologue and epilogue.

Sections below describe where arguments appear on the stack. The examples are written as if the function prologue described above had been used.

Position-independent code uses the <code>%ebx</code> register to hold the address of the global offset table. If a function needs the global offset table's address, either directly or indirectly, it is responsible for computing the value. See "Coding Examples" later in this chapter and "Dynamic Linking" in Chapter 5 for more information.

Integral and Pointer Arguments

As mentioned, a function receives all its arguments through the stack; the last argument is pushed first. In the standard calling sequence, the first argument is at offset 8(%ebp), the second argument is at offset 12(%ebp), and so on. Functions pass all integer-valued arguments as words, expanding or padding signed or unsigned bytes and halfwords as needed.

Figure 3-21: Integral and Pointer Arguments

Call	Argument	Stack address
	1	8(%ebp)
g(1, 2, 3,	2	12(%ebp)
(void *)0);	3	16(%ebp)
_	(void *)0	20(%ebp)

Floating-Point Arguments

The stack also holds floating-point arguments: single-precision values use one word, double-precision use two, and extended-precision use three. See "Coding Examples" for information about floating-point arguments and variable argument lists. The example below uses only double-precision arguments. Single- and extended-precision arguments behave as specified above.

Function Calling Sequence

Figure 3-22: Floating-Point Arguments

Call	Argument	Stack address
	word 0, 1.414	8(%ebp)
h(1.414, 1,	word 1, 1.414	12(%ebp)
2.998e10);	1	16(%ebp)
2.990e10),	word 0, 2.998e10	20(%ebp)
	word 1, 2.998e10	24(%ebp)

NOTE

The Intel386 architecture does not require doubleword alignment for doubleprecision values. Nevertheless, for data structure compatibility with other Intel architectures, compilers may provide a method to align double-precision values on doubleword boundaries.



CAUTION A compiler that provides the doubleword alignment mentioned above would have to maintain doubleword alignment for the stack. Moreover, the arguments in the preceding example would appear in different positions. Programs built with the doubleword alignment facility would not conform to the Intel386 ABI, and their function calling sequence would not be compatible with conforming Intel386 programs.

Structure and Union Arguments

As described in the data representation section, structures and unions can have byte, halfword, or word alignment, depending on the constituents. An argument's size is increased, if necessary, to make it a multiple of words. This may require tail padding, depending on the size of the argument. To ensure that data in the stack is properly aligned, the stack pointer should always point to a word boundary. Structure and union arguments are pushed onto the stack in the same manner as integral arguments, described above. This provides call-by-value semantics, letting the called function modify its arguments without affecting the calling function's object.

LOW-LEVEL SYSTEM INFORMATION

Figure 3-23: Structure and Union Arguments

Call	Argument	Callee	
i(1, s);	word 0, s word 1, s	8(%ebp) 12(%ebp) 16(%ebp)	

Operating System Interface

Virtual Address Space

Processes execute in a 32-bit virtual address space. Memory management translates virtual addresses to physical addresses, hiding physical addressing and letting a process run anywhere in the system's real memory. Processes typically begin with three logical segments, commonly called text, data, and stack. As Chapter 5 describes, dynamic linking creates more segments during execution, and a process can create additional segments for itself with system services.

Page Size

Memory is organized by pages, which are the system's smallest units of memory allocation. Page size can vary from one system to another, depending on the processor, memory management unit and system configuration. Processes may call sysconf (BA_OS) to determine the system's current page size.

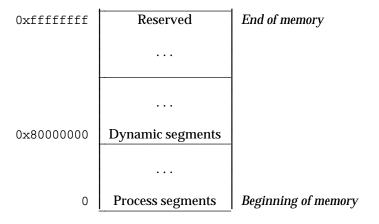
Virtual Address Assignments

Conceptually, processes have the full 32-bit address space available. In practice, however, several factors limit the size of a process.

- The system reserves a configuration-dependent amount of virtual space.
- The system reserves a configuration dependent amount of space per pro-
- A process whose size exceeds the system's available, combined physical memory and secondary storage cannot run. Although some physical memory must be present to run any process, the system can execute processes that are bigger than physical memory, paging them to and from secondary storage. Nonetheless, both physical memory and secondary storage are shared resources. System load, which can vary from one program execution to the next, affects the available amounts.

LOW-LEVEL SYSTEM INFORMATION







Programs that dereference null pointers are erroneous. although an implementation is not obliged to detect such erroneous behavior. Such programs may or may not fail on a particular system. To enhance portability, programmers are strongly cautioned not to rely on this behavior.

Process segments

Processes' loadable segments and stack may begin at 0. The exact addresses depend on the executable file format [see further information below and in Chapters 4 and 5]. Processes can control the amount of virtual memory allotted for stack space, as described below.

Dynamic segments

A process's dynamic segments reside below the reserved area.

Reserved

A reserved area resides at the top of virtual space.

As the figure shows, the system reserves the high end of virtual address space, with a process's dynamic segments below that. Although the exact boundary between the reserved area and a process depends on the system's configuration, the reserved area shall not consume more than 1 GB of the address space. Thus the user virtual address range has a minimum upper bound of 0×00000000 . Individual systems may reserve less space, increasing processes' virtual memory range.

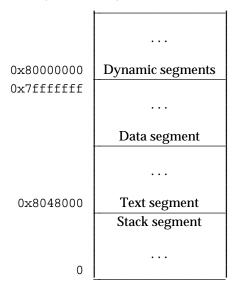
Operating System Interface

3-21

G

Although applications may control their memory assignments, the typical arrangement appears below.

Figure 3-25: Conventional Segment Arrangements



The process's text segment resides at 0x8048000. The data segment follows immediately, and dynamic segments occupy the higher range. When applications let the system choose addresses for dynamic segments (including shared object segments), it chooses high addresses. This leaves the ''middle'' of the address spectrum available for dynamic memory allocation with facilities such as malloc(BA_OS). Processes should *not* depend on finding their dynamic segments at particular virtual addresses. Facilities exist to let the system choose dynamic segment virtual addresses. The stack resides immediately below the text segment, growing toward lower addresses. This arrangement provides a little over 128 MB for the stack and about 2 GB for text and data.

Managing the Process Stack

Section "Process Initialization" in this chapter describes the initial stack contents. Stack addresses can change from one system to the next—even from one process execution to the next on the same system. Processes, therefore, should *not* depend on finding their stack at a particular virtual address.

LOW-LEVEL SYSTEM INFORMATION

A tunable configuration parameter controls the system maximum stack size. A process also can use setrlimit (BA_OS), to set its own maximum stack size, up to the system limit. On the Intel386, the stack segment has read and write permissions.

Coding Guidelines

Operating system facilities, such as mmap (KE_OS), allow a process to establish address mappings in two ways. First, the program can let the system choose an address. Second, the program can force the system to use an address the program supplies. This second alternative can cause application portability problems, because the requested address might not always be available. Differences in virtual address space can be particularly troublesome between different architectures, but the same problems can arise within a single architecture.

Processes' address spaces typically have three segment areas that can change size from one execution to the next: the stack [through setrlimit(BA_OS)], the data segment [through malloc(BA_OS)], and the dynamic segment area [through mmap(KE_OS)]. Changes in one area may affect the virtual addresses available for another. Consequently, an address that is available in one process execution might not be available in the next. A program that used mmap(KE_OS) to request a mapping at a specific address thus could appear to work in some environments and fail in others. For this reason, programs that wish to establish a mapping in their address space should let the system choose the address.

Despite these warnings about requesting specific addresses, the facility is both useful and can be used in a controlled manner. For example, a multiprocess application might map several files into the address space of each process and build relative pointers among the files' data. This could be done by having each process ask for a certain amount of memory at an address chosen by the system. After each process receives its own, private address from the system, it would map the desired files into memory, at specific addresses within the original area. This collection of mappings could be at different addresses in each process but their *relative* positions would be fixed. Without the ability to ask for specific addresses, the application could not build shared data structures, because the relative positions for files in each process would be unpredictable.

Operating System Interface

Processor Execution Modes

Four execution modes exist in the Intel386 architecture: ring 3 (or user mode) and three privileged rings. User processes run in user mode ring (the least privileged). The operating system kernel runs in a privileged mode ring, although the ABI does not specify which one. A program executes the lcall instruction through a system call gate to change execution modes, and thus the lcall instruction provides the low-level interface to system calls. For the Intel386, one low-level interface is defined: _exit(BA OS).

To ensure a process has a way to terminate itself, the system treats _exit as a special case. The ABI does not specify the implementation of other system services. Instead, programs should use the system libraries that Chapter 6 describes. Programs with other embedded lcall instructions do not conform to the ABI.

Figure 3-26: _exit System Trap

```
.globl_exit
_exit:
_movl $1, %eax
_lcall $7, $0
```

Exception Interface

As the Intel386 architecture manuals describe, the processor changes mode to handle *exceptions*, which may be synchronous, floating-point/coprocessor, or asynchronous. Synchronous and floating-point/coprocessor exceptions, being caused by instruction execution, can be explicitly generated by a process. This section, therefore, specifies those exception types with defined behavior. The Intel386 architecture classifies exceptions as *faults*, *traps*, and *aborts*. See the Intel *80386 Programmer's Reference Manual* for more information about their differences.

Hardware Exception Types

The operating system defines the following correspondence between hardware exceptions and the signals specified by signal (BA_OS).

LOW-LEVEL SYSTEM INFORMATION

Figure 3-27: Hardware Exceptions and Signals

Number	Exception Name	Signal
0	divide error fault	SIGFPE
1	single step trap/fault	SIGTRAP
2	nonmaskable interrupt	none
3	breakpoint trap	SIGTRAP
4	overflow trap	SIGSEGV
5	bounds check fault	SIGSEGV
6	invalid opcode fault	SIGILL
7	no coprocessor fault	SIGFPE
8	double fault abort	none
9	coprocessor overrun abort	SIGSEGV
10	invalid TSS fault	none
11	segment not present fault	none
12	stack exception fault	SIGSEGV
13	general protection fault/abort	SIGSEGV
14	page fault	SIGSEGV
15	(reserved)	
16	coprocessor error fault	SIGFPE
other	(unspecified)	SIGILL

Floating-point instructions exist in the architecture, but they may be implemented either in hardware (via the Intel387 chip) or in software (via the Intel387 emulator). In the case of "no coprocessor" exception, if the Intel387 emulator is configured into the kernel, the process receives no signal. Instead, the system intercepts the exception, emulates the instruction, and returns control to the process. A process receives SIGFPE for the "no coprocessor" exception only when the indicated floating-point instruction is illegal (invalid operands, and so on).

Software Trap Types

Because the int instruction generates traps, some hardware exceptions can be generated by software. However, the int instruction generates only traps and not faults; so it is not possible to match the exact hardware generated faults in software.

Operating System Interface

Process Initialization

This section describes the machine state that <code>exec(BA_OS)</code> creates for "infant" processes, including argument passing, register usage, stack frame layout, and so on. Programming language systems use this initial program state to establish a standard environment for their application programs. As an example, a C program begins executing at a function named <code>main</code>, conventionally declared in the following way.

Figure 3-28: Declaration for main

```
extern int main(int argc, char *argv[], char *envp[]);
```

Briefly, argc is a non-negative argument count; argv is an array of argument strings, with argv[argc]==0; and envp is an array of environment strings, also terminated by a null pointer.

Although this section does not describe C program initialization, it gives the information necessary to implement the call to main or to the entry point for a program in any other language.

Special Registers

As the Intel386 architecture defines, several state registers control and monitor the processor: the Machine Status Word register (MSW, also known as register %cr0), EFLAGS register, the floating-point status register, and the floating-point control register. Application programs cannot access the full EFLAGS register directly; because they run in the processor's *user mode*, and the instructions to write some of the bits of the EFLAGS register are privileged. Nonetheless, a program has access to many of the flags in the EFLAGS register. Flags identified with an "*" below are not modifiable by a user mode process, they either have unspecified values or do not affect user program behavior. At process initialization, the EFLAGS register contains the following values.

LOW-LEVEL SYSTEM INFORMATION

Figure 3-29: EFLAGS Register Fields

Flag	Value	Note
CF	unspecified	Carry flag
PF	unspecified	Parity flag
AF	unspecified	Auxiliary carry flag
ZF	unspecified	Zero flag
SF	unspecified	Sign flag
TF	unspecified	Trap flag
IF*	unspecified	Interrupt enable
DF	0	Direction flag low to high
OF	unspecified	Overflow flag
IOPL*	unspecified	I/O privilege level
NT*	unspecified	Nested task
RF*	unspecified	Resume flag
VM*	unspecified	Virtual 8086 mode

The Intel386 architecture defines floating-point instructions, and those instructions work whether the processor has a hardware floating-point unit or not. (A system may provide hardware or software floating-point facilities.) Consequently, the contents of the MSW register is not specified, letting the system set it according to the hardware configuration. In any case, however, the processor presents a working floating-point implementation, including the Intel387 status and control word registers with the following values at process initialization.

Figure 3-30: Floating-Point Control Word

Field	Value	Note	
IC	1	Affine infinity (for compatibility)	
RC	00	Round to nearest or even	
PC	11	53-bit (double precision)	G
PM	1	Precision masked	
UM	1	Underflow masked	
OM	1	Overflow	G
ZM	1	Zero divide	G
DM	1	Denormalized operand masked	
IM	1	Invalid operation	G

Operating System Interface



CAUTION/ The initial floating-point state should be changed with care. In particular, many floating-point routines may produce undefined behavior if the precision control is set to less than 53 bits. The fpstart routine (see Chapter 6) changes the precision control to 64 bits and sets all exceptions to be asked. This is the default state required for conformance to the ANSI C standard and to the IEEE 754 Floating-point standard.

- G G G
- Ğ

Process Stack and Registers

When a process receives control, its stack holds the arguments and environment from $exec(BA_OS)$.

Figure 3-31: Initial Process Stack

Unspecified	High addresses
Information block, including	
argument strings,	
environment strings,	
auxiliary information	
(size varies)	
Unspecified	
Null auxiliary vector entry	
Auxiliary vector	
(2-word entries)	
0 word	
Environment pointers	
(one word each)	
0 word	
Argument pointers	
(Argument count words)	
Argument count	
Undefined	Low addresses

4(%esp) 0(%esp)

LOW-LEVEL SYSTEM INFORMATION

Argument strings, environment strings, and the auxiliary information appear in no specific order within the information block; the system makes no guarantees about their arrangement. The system also may leave an unspecified amount of memory between the null auxiliary vector entry and the beginning of the information block.

General and floating-point register values are unspecified at process entry, with the exceptions appearing below. Consequently, a program that requires registers to have specific values must set them explicitly during process initialization. It should *not* rely on the operating system to set all registers to 0.

%ebp The content of this register is unspecified at process initialization

time, but the user code should mark the deepest stack frame by setting the frame pointer to zero. No other frame's *ebp should

have a zero value.

Lesp Performing its usual job, the stack pointer holds the address of the

bottom of the stack, which is guaranteed to be word aligned.

%edx In a conforming program, this register contains a function pointer

that the application should register with atexit(BA_OS). This function is used for shared object termination code [see "Dynamic

Linking" in Chapter 5 of the *System V ABI*].

%cs, %ds, %es, %ss

The segment registers are initialized so that the user process can address the code, data, and stack segments using a 32-bit virtual address. A program that alters their values does not conform to the ABI and has undefined behavior.

Every process has a stack, but the system defines *no* fixed stack address. Furthermore, a program's stack address can change from one system to another—even from one process invocation to another. Thus the process initialization code must use the stack address in <code>%esp</code>. Data in the stack segment at addresses below the stack pointer contain undefined values.

Whereas the argument and environment vectors transmit information from one application program to another, the auxiliary vector conveys information from the operating system to the program. This vector is an array of the following structures, interpreted according to the a_type member.

Operating System Interface

Figure 3-32: Auxiliary Vector

```
typedef struct
{
    int a_type;
    union {
        long a_val;
        void *a_ptr;
        void (*a_fcn)();
    } a_un;
} auxv_t;
```

Figure 3-33: Auxiliary Vector Types, a_type

Name	Value	a_un	
AT_NULL	0	ignored	
AT_IGNORE	1	ignored	
AT_EXECFD	2	a_val	
AT_PHDR	3	a_ptr	
AT_PHENT	4	a_val	
AT_PHNUM	5	a_val	
AT_PAGESZ	6	a_val	
AT_BASE	7	a_ptr	
AT_FLAGS	8	a_val	
AT_ENTRY	9	a_ptr	
AT_LIBPATH	10	a_val	M
AT_FPHW	11	a_val	M
AT_INTP_DEVICE	12	a_val	M
AT_INTP_INODE	13	a_val	M

AT_NULL The auxiliary vector has no fixed length; instead its last entry's a_type member has this value.

LOW-LEVEL SYSTEM INFORMATION

This type indicates the entry has no meaning. The corresponding AT_IGNORE value of a un is undefined. As Chapter 5 describes, exec(BA OS) may pass control to an AT EXECFD interpreter program. When this happens, the system places either an entry of type AT_EXECFD or one of type AT_PHDR in the auxiliary vector. The entry for type AT_EXECFD uses the a_val member to contain a file descriptor open to read the application program's object file. AT_PHDR Under some conditions, the system creates the memory image of the application program before passing control to the interpreter program. When this happens, the a_ptr member of the AT_PHDR entry tells the interpreter where to find the program header table in the memory image. If the AT PHDR entry is present, entries of types AT_PHENT, AT_PHNUM, and AT_ENTRY must also be present. See Chapter 5 in both the System V ABI and the processor supplement for more information about the program header table. AT_PHENT The a_val member of this entry holds the size, in bytes, of one entry in the program header table to which the AT_PHDR entry points. The a_val member of this entry holds the number of entries in AT_PHNUM the program header table to which the AT_PHDR entry points. If present, this entry's a_val member gives the system page size, AT PAGESZ in bytes. The same information also is available through sysconf (BA OS). AT_BASE The a_ptr member of this entry holds the base address at which the interpreter program was loaded into memory. See "Program Header" in the System V ABI for more information about the base address. AT FLAGS If present, the a_val member of this entry holds one-bit flags. Bits with undefined semantics are set to zero. AT ENTRY The a_ptr member of this entry holds the entry point of the application program to which the interpreter program should transfer control. AT LIBPATH The a_val member of this entry is non-zero if the dynamic linker should examine LD LIBRARY PATH when searching for shared objects of the process based on the security considerations in the Shared Object Dependency section in Chapter 5 of the gABI.

Operating System Interface

AT_FPHW The a_val member of this entry will be set to

Figure 3-34: AT FPHW values

Value	Meaning
0	if no floating point support exists
1	if floating point software emulation exists
2	if it has a 80287 chip if it has a 80387 chip or a 80487 chip
3	if it has a 80387 chip or a 80487 chip

AT_INTP_DEVICE

The a_val member of this entry holds the device number of the file from which the dynamic linker is loaded.

AT_INTP_INODE

The a_val member of this entry holds the inode of the file from which the dynamic linker is loaded.

Other auxiliary vector types are reserved. No flags are currently defined for AT_FLAGS, on the Intel386 architecture.

To illustrate, suppose an example process receives two arguments.

- echo
- abi

It also inherits two environment strings (this example is not intended to show a fully configured execution environment).

- HOME=/home/dir
- PATH=/usr/bin:

Its one non-null auxiliary vector entry holds a file descriptor.

■ {AT_EXECFD, 13}

The resulting stack resides below 0x8048000, growing toward lower addresses.

LOW-LEVEL SYSTEM INFORMATION

Figure 3-35: Example Process Stack

		n	:	\0	pad	High addresses
		r	/	b	i	
		=	/	u	s	
	0x8047ff0	Р	A	Т	Н	
		d	i	r	\0	
		0	m	е	/	
		Е	=	/	h	
	0x8047fe0	\0	Н	0	М	
		\0	a	b	i	
		е	С	h	0	
			()		
	0x8047fd0		()		
		13				
			2	2		Auxiliary vector
		0				
	0x8047fc0	0x8047ff0				
		0x8047fe1				Environment vector
		0				
		0x8047fdd				
	0x8047fb0	0x8047fd8			Argument vector	
0(%esp),	0x8047fac			2		Argument count
		Undefined			Low addresses	

Coding Examples

This section discusses example code sequences for fundamental operations such as calling functions, accessing static objects, and transferring control from one part of a program to another. Previous sections discuss how a program may use the machine or the operating system, and they specify what a program may and may not assume about the execution environment. Unlike previous material, the information here illustrates how operations *may* be done, not how they *must* be done.

As before, examples use the ANSI C language. Other programming languages may use the same conventions displayed below, but failure to do so does *not* prevent a program from conforming to the ABI. Two main object code models are available.

- Absolute code. Instructions can hold absolute addresses under this model. To execute properly, the program must be loaded at a specific virtual address, making the program's absolute addresses coincide with the process's virtual addresses.
- *Position-independent code*. Instructions under this model hold relative addresses, *not* absolute addresses. Consequently, the code is not tied to a specific load address, allowing it to execute properly at various positions in virtual memory.

Following sections describe the differences between these models. Code sequences for the models (when different) appear together, allowing easier comparison.



Examples below show code fragments with various simplifications. They are intended to explain addressing modes, not to show optimal code sequences nor to reproduce compiler output.



When other sections of this document show assembly language code sequences, they typically show only the absolute versions. Information in this section explains how position-independent code would alter the examples.

LOW-LEVEL SYSTEM INFORMATION

Code Model Overview

When the system creates a process image, the executable file portion of the process has fixed addresses, and the system chooses shared object library virtual addresses to avoid conflicts with other segments in the process. To maximize text sharing, shared objects conventionally use position-independent code, in which instructions contain no absolute addresses. Shared object text segments can be loaded at various virtual addresses without having to change the segment images. Thus multiple processes can share a single shared object text segment, even though the segment resides at a different virtual address in each process.

Position-independent code relies on two techniques.

- Control transfer instructions hold offsets relative to the extended instruction pointer (EIP). An EIP-relative branch or function call computes its destination address in terms of the current instruction pointer, *not* relative to any absolute address.
- When the program requires an absolute address, it computes the desired value. Instead of embedding absolute addresses in the instructions, the compiler generates code to calculate an absolute address during execution.

Because the Intel386 architecture provides EIP-relative call and branch instructions, compilers can satisfy the first condition easily.

A *global offset table* provides information for address calculation. Position-independent object files (executable and shared object files) have this table in their data segment. When the system creates the memory image for an object file, the table entries are relocated to reflect the absolute virtual addresses as assigned for an individual process. Because data segments are private for each process, the table entries can change—unlike text segments, which multiple processes share.

Assembly language examples below show the explicit notation needed for position-independent code.

name@GOT(%ebx)

This expression denotes an *ebx-relative reference to the global offset table entry for the symbol *name*. The *ebx register contains the absolute address of the global offset table, as explained below.

name@GOTOFF(%ebx)

This expression denotes an %ebx-relative reference to the symbol name. Again, %ebx holds the global offset table address. Note this expression references name, not the global offset table entry for name.

Coding Examples 3-35

name@PLT This expression denotes an EIP-relative reference to the procedure linkage table entry for the symbol name.

GLOBAL OFFSET TABLE

The symbol _GLOBAL_OFFSET_TABLE_ is used to access the global offset table. When an instruction uses the symbol, it sees the offset between the current instruction and the global offset table as the symbol value.

Position-Independent Function Prologue

This section describes the function prologue for position-independent code. A function's prologue allocates the local stack space, saves any registers it must preserve, and sets register <code>%ebx</code> to the global offset table's address. Because <code>%ebx</code> is private for each function and preserved across function calls, a function calculates its value once at the entry.

Figure 3-36: Calculating Global Offset Table Address

Line			Code
1	T1.	call	.L1
3	.L1:	popl addl	<pre>\$_GLOBAL_OFFSET_TABLE_+[L1], %ebx</pre>

These three lines accomplish the following.

- 1. The call instruction pushes the *absolute* address of the next instruction onto the stack.
- 2. Consequently, the popl instruction pops the absolute address of .L1 into register $\texttt{\ensuremath{\$}ebx}.$
- 3. The last instruction computes the desired absolute value into %ebx. This works because _GLOBAL_OFFSET_TABLE_ in the expression gives the distance from the addl instruction to the global offset table; [.-.L1] gives the distance from .L1 to the addl instruction. Adding their sum to the absolute address of .L1, already in %ebx, gives the absolute address of the global offset table.

LOW-LEVEL SYSTEM INFORMATION

This computation can be added to the standard function prologue, giving the standard prologue for position-independent code. To illustrate, the following function prologue allocates 80 bytes of local stack space and saves the local registers %ebx, %esi, and %edi.

Figure 3-37: Position-Independent Function Prologue

```
prologue:
    pushl %ebp
    movl %esp, %ebp
    subl $80, %esp
    pushl %edi
    pushl %esi
    pushl %ebx
    call .Ll
.L1: popl %ebx
    addl $_GLOBAL_OFFSET_TABLE_+[.-.L1], %ebx
```

Position-independent and absolute code use the same function epilogue.

Data Objects

This discussion excludes stack-resident objects, because programs always compute their virtual addresses relative to the stack and frame pointers. Instead, this section describes objects with static storage duration.

In the Intel386 architecture, all memory reference instructions can address any location within the 32-bit address space. Symbolic references in absolute code put the symbols' values—or absolute virtual addresses—into instructions.

Coding Examples 3-37

Figure 3-38: Absolute Data Access

extern int src; extern int dst; extern int *ptr; ptr = &dst; *ptr = src;

Assembly				
.globl	src, dst, ptr			
movl	\$dst, ptr			
movl	ptr, %eax			
movl	src, %edx			
movl	%edx, (%eax)			

Position-independent instructions cannot contain absolute addresses. Instead, instructions that reference symbols hold the symbols' offsets into the global offset table. Combining the offset with the global offset table address in %ebx gives the absolute address of the table entry holding the desired address.

Figure 3-39: Position-Independent Data Access

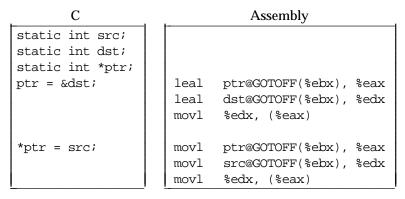
extern int src; extern int dst; extern int *ptr; ptr = &dst; *ptr = src;

Assembly				
.globl	src, dst, ptr			
movl movl	<pre>ptr@GOT(%ebx), %eax dst@GOT(%ebx), %edx %edx, (%eax)</pre>			
movl movl	ptr@GOT(%ebx), %eax (%eax), %eax			
movl	src@GOT(%ebx), %edx			
movl	(%edx), %edx			
movl	%edx, (%eax)			

Finally, position-independent references to static data may be optimized. Because %ebx holds a known address, the global offset table, a program may use it as a base register. External references should use the global offset table entry, because dynamic linking may bind the entry to a definition outside the current object file's scope.

LOW-LEVEL SYSTEM INFORMATION

Figure 3-40: Position-Independent Static Data Access



Function Calls

Programs use the call instruction to make direct function calls. A call instruction's destination is an EIP-relative value that can reach any address in the 32-bit virtual space. Even when the code for a function resides in a shared object, the caller uses the same assembly language instruction sequence, although in that case control passes from the original call, through an indirection sequence, to the desired destination. See "Procedure Linkage Table" in Chapter 5 for more information on the indirection sequence.

Figure 3-41: Absolute Direct Function Call

. C	Ass	embly
extern void function();	.globl	function
function();	call	function

Dynamic linking may redirect a function call outside the current object file's scope; so position-independent calls should use the procedure linkage table explicitly.

Coding Examples 3-39

Figure 3-42: Position-Independent Direct Function Call

C	 Assembly		
<pre>extern void function();</pre>	.globl	function	
function();	call	function@PLT	

Indirect function calls use the indirect call instruction.

Figure 3-43: Absolute Indirect Function Call

For position-independent code, the global offset table supplies absolute addresses for all required symbols, whether the symbols name objects or functions.

Figure 3-44: Position-Independent Indirect Function Call

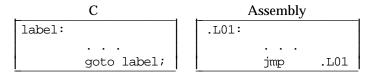
C Assembly extern void (*ptr)(); .globl ptr, name extern void name(); ptr = name; movl ptr@GOT(%ebx), %eax movl name@GOT(%ebx), %edx movl %edx, (%eax) (*ptr)(); movlptr@GOT(%ebx), %eax call *(%eax)

LOW-LEVEL SYSTEM INFORMATION

Branching

Programs use branch instructions to control their execution flow. As defined by the Intel386 architecture, branch instructions hold an EIP-relative value with a signed 32-bit range, allowing a jump to any location within the virtual address space.

Figure 3-45: Branch Instruction, All Models



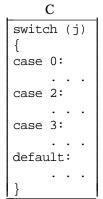
C switch statements provide multiway selection. When the case labels of a switch statement satisfy grouping constraints, the compiler implements the selection with an address table. The following examples use several simplifying conventions to hide irrelevant details:

- The selection expression resides in register %eax;
- case label constants begin at zero;
- case labels, default, and the address table use assembly names .Lcasei, .Ldef, and .Ltab, respectively.

Address table entries for absolute code contain virtual addresses; the selection code extracts an entry's value and jumps to that address. Position-independent table entries hold offsets; the selection code computes a destination's absolute address.

Coding Examples 3-41

Figure 3-46: Absolute switch Code



Assembly cmpl \$3, %eax ja .Ldef *.Ltab(,%eax,4) jmp .Ltab: .long .Lcase0 .Ldef .long .long .Lcase2 .long .Lcase3

Figure 3-47: Position-Independent switch Code

```
Assembly
             $3, %eax
    cmpl
     ja
             .Ldef
             .Ltab@GOTOFF(%ebx), %edx
     leal
    movl
             (%edx,%eax,4), %eax
    movl
             .Ltab@GOTOFF(%ebx,%eax,4), %eax
             .Ljmp
    call
.Ljmp:
             %ecx
    popl
     addl
             %ecx, %eax
     jmp
             *%eax
.Ltab:
             .Lcase0 - .Ljmp
     .long
             .Ldef - .Ljmp
     .long
     .long
             .Lcase2 - .Ljmp
             .Lcase3 - .Ljmp
     .long
```

LOW-LEVEL SYSTEM INFORMATION

C Stack Frame

Figure 3-48 shows the C stack frame organization. It conforms to the standard stack frame with designated roles for unspecified areas in the standard frame.

M This represents one possible organization of the C stack frame. Usage of %ebp as a frame pointer, the exact positions of the callee saved registers, and space for local storage is implementation specific.

Figure 3-48: C Stack Frame

Base	Offset	Contents	
%ebp	4n+8	argument word <i>n</i>	High addresses
	8	argument word 0	
	4	return address	
%ebp	0	caller's %ebp	
%ebp	-4	x words local space:	
		automatic variables,	
		temporaries,	
%ebp	-4x	etc.	
%esp	12	-	
%esp	8	caller's %edi	
	4	caller's %esi	
%esp	0	caller's %ebx	Low addresses

A C stack frame doesn't normally change size during execution. The exception is dynamically allocated stack memory, discussed below. By convention, a function allocates automatic (local) variables in the middle of its frame and references them as negative offsets from <code>%ebp</code>. Its incoming arguments reside in the previous frame, referenced as positive offsets from <code>%ebp</code>. If necessary, a function saves the values of <code>%edi</code>, <code>%esi</code>, and <code>%ebx</code> in the positions shown and restores their values before returning to the caller. The positions may be different from the diagram above, depending on which of these three registers the function saves and restores.

Coding Examples 3-43

Variable Argument List

Previous sections describe the rules for passing arguments. Unfortunately, some otherwise portable C programs depend on the argument passing scheme, implicitly assuming that 1) all arguments reside on the stack, and 2) arguments appear in increasing order on the stack. Programs that make these assumptions never have been portable, but they have worked on many machines, including the Intel386. Nonetheless, portable C programs should use the facilities defined in the header files <stdarg.h> or <varags.h> to deal with variable argument lists.

Allocating Stack Space Dynamically

Unlike some other languages, C does not need dynamic stack allocation *within* a stack frame. Frames are allocated dynamically on the program stack, depending on program execution, but individual stack frames can have static sizes. Nonetheless, the architecture supports dynamic allocation for those languages that require it, and the standard calling sequence and stack frame support it as well. Thus languages that need dynamic stack frame sizes can call C functions, and vice versa

Figure 3-48 shows the layout of the C stack frame. The double line divides the area referenced from <code>%ebp</code> from the area referenced from <code>%ebp</code>. Dynamic space is allocated below the line, as a downward growing heap whose size changes as required. Typical C functions have no space in the heap. All areas above the heap in the current frame have a known size to the compiler. Dynamic stack allocation thus takes the following steps.

- 1. Stack frames are word aligned; dynamic allocation should preserve this property. Thus the program rounds (up) the desired byte count to a multiple of 4.
- 2. The program decreases the stack pointer by the rounded byte count, increasing its frame size. At this point, the "new" space resides just below the register save area at the bottom of the stack.
- 3. The program copies the register save area (three or fewer words) to the bottom of the stack, effectively moving the new space up into the frame.

LOW-LEVEL SYSTEM INFORMATION

The register save area is reserved and should not be used for purposes outside of this document.

G

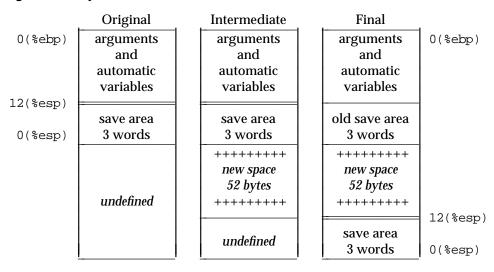
Even in the presence of signals, dynamic allocation is "safe." If a signal interrupts allocation, one of three things can happen.

- The signal handler can return. The process then resumes the dynamic allocation from the point of interruption.
- The signal handler can execute a non-local goto, or longjmp [see setjmp(BA_LIB)]. This resets the process to a new context in a previous stack frame, automatically discarding the dynamic allocation.
- The process can terminate.

Regardless of when the signal arrives during dynamic allocation, the result is a consistent (though possibly dead) process.

To illustrate, assume a program wants to allocate 50 bytes, and it has saved three registers in the bottom of the frame. The first step is rounding 50 to 52, making it a multiple of 4. Figure 3-49 shows how the stack frame changes.

Figure 3-49: Dynamic Stack Allocation



New space starts at 12(%esp). As described, every dynamic allocation in *this* function will return a new area starting at 12(%esp), leaving previous heap objects untouched (other functions could have different heap addresses).

Coding Examples 3-45

Consequently, the compiler should compute the absolute address for each area, avoiding relative references. Otherwise, future allocations in the same frame would destroy the heap's integrity.

Existing stack objects reside at fixed offsets from the frame pointer (%ebp). Dynamic allocation preserves those offsets, because the frame pointer does not change and the objects relative to it do not move. Objects relative to the stack pointer (%esp) move, but their %esp-relative positions do not change. Accordingly, compilers arrange not to publicize the absolute address of any object in the bottom half of the stack frame (in a way that violates the scope rules). %esp-relative references stay valid after dynamic allocation, but absolute addresses do not.

No special code is needed to free dynamically allocated stack memory. The function return resets the stack pointer and removes the entire stack frame, including the heap, from the stack. Naturally, a program should not reference heap objects after they have gone out of scope.

LOW-LEVEL SYSTEM INFORMATION

4 OBJECT FILES

ELF Header Machine Information	4-1 4-1
Sections	4-2
Special Sections	4-2
Symbol Table	4-3
Symbol Values	4-3
Relocation	4-4
Relocation Types	4-4

Table of Contents

ELF Header

Machine Information

For file identification in e_ident, the Intel386 architecture requires the following values.

Figure 4-1: Intel386 Identification, e_ident

Position	Value
e_ident[EI_CLASS] e_ident[EI_DATA]	ELFCLASS32 ELFDATA2LSB
e_ident[Ei_DAIA]	FLFDATAZLOB

Processor identification resides in the ELF header's e_machine member and must have the value ${\tt EM_386}$.

The ELF header's e_flags member holds bit flags associated with the file. The Intel386 architecture defines no flags; so this member contains zero.

ELF Header 4-1

Sections

Special Sections

Various sections hold program and control information. Sections in the list below are used by the system and have the indicated types and attributes.

Figure 4-2: Special Sections

Name	Type	Attributes
.got	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
.plt	SHT_PROGBITS	SHF_ALLOC + SHF_EXECINSTR

- .got This section holds the global offset table. See "Coding Examples" in Chapter 3 and "Global Offset Table" in Chapter 5 for more information.
- .plt This section holds the procedure linkage table. See "Procedure Linkage Table" in Chapter 5 for more information.

4-2 OBJECT FILES

Symbol Table

Symbol Values

If an executable file contains a reference to a function defined in one of its associated shared objects, the symbol table section for that file will contain an entry for that symbol. The st_shndx member of that symbol table entry contains SHN_UNDEF. This signals to the dynamic linker that the symbol definition for that function is not contained in the executable file itself. If that symbol has been allocated a procedure linkage table entry in the executable file, and the st_value member for that symbol table entry is non-zero, the value will contain the virtual address of the first instruction of that procedure linkage table entry. Otherwise, the st_value member contains zero. This procedure linkage table entry address is used by the dynamic linker in resolving references to the address of the function. See "Function Addresses" in Chapter 5 for details.

Symbol Table 4-3

Relocation

Relocation Types

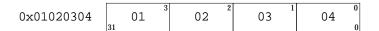
Relocation entries describe how to alter the following instruction and data fields (bit numbers appear in the lower box corners).



word32

word32

This specifies a 32-bit field occupying 4 bytes with arbitrary byte alignment. These values use the same byte order as other word values in the Intel386 architecture.



Calculations below assume the actions are transforming a relocatable file into either an executable or a shared object file. Conceptually, the link editor merges one or more relocatable files to form the output. It first decides how to combine and locate the input files, then updates the symbol values, and finally performs the relocation. Relocations applied to executable or shared object files are similar and accomplish the same result. Descriptions below use the following notation.

- A This means the addend used to compute the value of the relocatable field.
- B This means the base address at which a shared object has been loaded into memory during execution. Generally, a shared object file is built with a 0 base virtual address, but the execution address will be different. See "Program Header" in the System V ABI for more information about the base address.
- G This means the offset into the global offset table at which the address of the relocation entry's symbol will reside during execution. See "Coding Examples" in Chapter 3 and "Global Offset Table" in Chapter 5 for more information.

4-4 OBJECT FILES

- This means the address of the global offset table. See "Coding Examples" in Chapter 3 and "Global Offset Table" in Chapter 5 for more information.
- This means the place (section offset or address) of the procedure linkage table entry for a symbol. A procedure linkage table entry redirects a function call to the proper destination. The link editor builds the initial procedure linkage table, and the dynamic linker modifies the entries during execution. See "Procedure Linkage Table" in Chapter 5 for more information.
- This means the place (section offset or address) of the storage unit being relocated (computed using r_offset).
- This means the value of the symbol whose index resides in the relocation entry.

A relocation entry's r_{offset} value designates the offset or virtual address of the first byte of the affected storage unit. The relocation type specifies which bits to change and how to calculate their values. The Intel386 architecture uses only <code>Elf32_Rel</code> relocation entries, the field to be relocated holds the addend. In all cases, the addend and the computed result use the same byte order.

Figure 4-4: Relocation Types

Name	Value	Field	Calculation
R_386_NONE	0	none	none
R_386_32	1	word32	S + A
R_386_PC32	2	word32	S + A - P
R_386_GOT32	3	word32	G + A - P
R_386_PLT32	4	word32	L + A - P
R_386_COPY	5	none	none
R_386_GLOB_DAT	6	word32	S
R_386_JMP_SLOT	7	word32	S
R_386_RELATIVE	8	word32	B + A
R_386_GOTOFF	9	word32	S + A - GOT
R_386_GOTPC	10	word32	GOT + A - P
			1

Some relocation types have semantics beyond simple calculation.

R_386_GOT32

This relocation type computes the distance from the base of the global offset table to the symbol's global offset table entry. It additionally instructs the link editor to build a global offset table.

Relocation 4-5

R_386_PLT32 This relocation type computes the address of the symbol's procedure linkage table entry and additionally instructs the link editor to build a procedure linkage table. The link editor creates this relocation type for dynamic link-R_386_COPY ing. Its offset member refers to a location in a writable segment. The symbol table index specifies a symbol that should exist both in the current object file and in a shared object. During execution, the dynamic linker copies data associated with the shared object's symbol to the location specified by the offset. This relocation type is used to set a global offset table entry to R 386 GLOB DAT the address of the specified symbol. The special relocation type allows one to determine the correspondence between symbols and global offset table entries. R_386_JMP_SLOT The link editor creates this relocation type for dynamic linking. Its offset member gives the location of a procedure linkage table entry. The dynamic linker modifies the procedure linkage table entry to transfer control to the designated symbol's address [see "Procedure Linkage Table" in Chapter 5]. R_386_RELATIVE The link editor creates this relocation type for dynamic linking. Its offset member gives a location within a shared object that contains a value representing a relative address. The dynamic linker computes the corresponding virtual address by adding the virtual address at which the shared object was loaded to the relative address. Relocation entries for this type must specify 0 for the symbol table index. R 386 GOTOFF This relocation type computes the difference between a symbol's value and the address of the global offset table. It additionally instructs the link editor to build the global offset table. This relocation type resembles R_386_PC32, except it uses the R 386 GOTPC address of the global offset table in its calculation. The symbol referenced in this relocation normally is _GLOBAL_OFFSET_TABLE_, which additionally instructs the link editor to build the global offset table.

4-6 OBJECT FILES

5 PROGRAM LOADING AND DYNAMIC LINKING

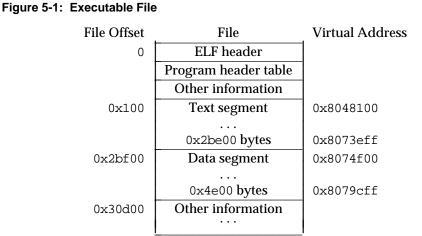
Program Loading	5-1
Dynamic Linking	5-5
Dynamic Section	5-5
Global Offset Table	5-5
Function Addresses	5-6
Procedure Linkage Table	5-7
Program Interpreter	5-10

Table of Contents

Program Loading

As the system creates or augments a process image, it logically copies a file's segment to a virtual memory segment. When—and if—the system physically reads the file depends on the program's execution behavior, system load, and so on. A process does not require a physical page unless it references the logical page during execution, and processes commonly leave many pages unreferenced. Therefore delaying physical reads frequently obviates them, improving system performance. To obtain this efficiency in practice, executable and shared object files must have segment images whose file offsets and virtual addresses are congruent, modulo the page size.

Virtual addresses and file offsets for the Intel386 architecture segments are congruent modulo 4 KB (0x1000) or larger powers of 2. Because 4 KB is the maximum page size, the files will be suitable for paging regardless of physical page size.



Program Loading 5-1

Figure 5-2: Program Header Segments

Member	Text	Data
p_type	PT_LOAD	PT_LOAD
p_offset	0x100	0x2bf00
p_vaddr	0x8048100	0x8074f00
p_paddr	unspecified	unspecified
p_filesz	0x2be00	0x4e00
p_memsz	0x2be00	0x5e24
p_flags	PF_R+PF_X	PF_R+PF_W+PF_X
p_align	0x1000	0x1000

Although the example's file offsets and virtual addresses are congruent modulo 4 KB for both text and data, up to four file pages hold impure text or data (depending on page size and file system block size).

- The first text page contains the ELF header, the program header table, and other information.
- The last text page holds a copy of the beginning of data.
- The first data page has a copy of the end of text.
- The last data page may contain file information not relevant to the running process.

Logically, the system enforces the memory permissions as if each segment were complete and separate; segments' addresses are adjusted to ensure each logical page in the address space has a single set of permissions. In the example above, the region of the file holding the end of text and the beginning of data will be mapped twice: at one virtual address for text and at a different virtual address for data.

The end of the data segment requires special handling for uninitialized data, which the system defines to begin with zero values. Thus if a file's last data page includes information not in the logical memory page, the extraneous data must be set to zero, not the unknown contents of the executable file. "Impurities" in the other three pages are not logically part of the process image; whether the system expunges them is unspecified. The memory image for this program follows, assuming $4~{\rm KB}~(0{\approx}1000)$ pages.

PROGRAM LOADING AND DYNAMIC LINKING

Figure 5-3: Process Image Segments

Virtual Address	Contents	Segment
0x8048000	<i>Header padding</i> 0x100 bytes	
0x8048100	Text segment	
		Text
	0x2be00 bytes	
0x8073f00	Data padding	
	0x100 bytes	
0x8074000	Text padding	
	0xf00 bytes	
0x8074f00	Data segment	
		ъ.
	0x4e00 bytes	Data
0x8079d00	Uninitialized data	
	0x1024 zero bytes	
0x807ad24	Page padding	
	0x2dc zero bytes	

One aspect of segment loading differs between executable files and shared objects. Executable file segments typically contain absolute code (see "Coding Examples" in Chapter 3). To let the process execute correctly, the segments must reside at the virtual addresses used to build the executable file. Thus the system uses the p_vaddr values unchanged as virtual addresses.

On the other hand, shared object segments typically contain position-independent code. This lets a segment's virtual address change from one process to another, without invalidating execution behavior. Though the system chooses virtual addresses for individual processes, it maintains the segments' *relative positions*. Because position-independent code uses relative addressing between segments, the difference between virtual addresses in memory must match the difference between virtual addresses in the file. The following table shows possible shared object virtual address assignments for several processes, illustrating constant relative positioning. The table also illustrates the base address computations.

Program Loading 5-3

Figure 5-4: Example Shared Object Segment Addresses

Source	Text	Data	Base Address
File	0x200	0x2a400	0x0
Process 1	0x80000200	0x8002a400	0x80000000
Process 2	0x80081200	0x800ab400	0x80081000
Process 3	0x900c0200	0x900ea400	0x900c0000
Process 4	0x900c6200	0x900f0400	0x900c6000

PROGRAM LOADING AND DYNAMIC LINKING

Dynamic Linking

Dynamic Section

Dynamic section entries give information to the dynamic linker. Some of this information is processor-specific, including the interpretation of some entries in the dynamic structure.

DT_PLTGOT

On the Intel386 architecture, this entry's d_ptr member gives the address of the first entry in the global offset table. As mentioned below, the first three global offset table entries are reserved, and two are used to hold procedure linkage table information.

Global Offset Table

Position-independent code cannot, in general, contain absolute virtual addresses. Global offset tables hold absolute addresses in private data, thus making the addresses available without compromising the position-independence and sharability of a program's text. A program references its global offset table using position-independent addressing and extracts absolute values, thus redirecting position-independent references to absolute locations.

Initially, the global offset table holds information as required by its relocation entries [see "Relocation" in Chapter 4]. After the system creates memory segments for a loadable object file, the dynamic linker processes the relocation entries, some of which will be type R_386_GLOB_DAT referring to the global offset table. The dynamic linker determines the associated symbol values, calculates their absolute addresses, and sets the appropriate memory table entries to the proper values. Although the absolute addresses are unknown when the link editor builds an object file, the dynamic linker knows the addresses of all memory segments and can thus calculate the absolute addresses of the symbols contained therein.

If a program requires direct access to the absolute address of a symbol, that symbol will have a global offset table entry. Because the executable file and shared objects have separate global offset tables, a symbol's address may appear in several tables. The dynamic linker processes all the global offset table relocations before giving control to any code in the process image, thus ensuring the absolute addresses are available during execution.

Dynamic Linking 5-5

The table's entry zero is reserved to hold the address of the dynamic structure, referenced with the symbol _DYNAMIC. This allows a program, such as the dynamic linker, to find its own dynamic structure without having yet processed its relocation entries. This is especially important for the dynamic linker, because it must initialize itself without relying on other programs to relocate its memory image. On the Intel386 architecture, entries one and two in the global offset table also are reserved. "Procedure Linkage Table" below describes them.

The system may choose different memory segment addresses for the same shared object in different programs; it may even choose different library addresses for different executions of the same program. Nonetheless, memory segments do not change addresses once the process image is established. As long as a process exists, its memory segments reside at fixed virtual addresses.

A global offset table's format and interpretation are processor-specific. For the Intel386 architecture, the symbol _GLOBAL_OFFSET_TABLE_ may be used to access the table.

Figure 5-5: Global Offset Table

extern Elf32_Addr __GLOBAL_OFFSET_TABLE_[];

The symbol _GLOBAL_OFFSET_TABLE_ may reside in the middle of the .got section, allowing both negative and non-negative "subscripts" into the array of addresses.

Function Addresses

References to the address of a function from an executable file and the shared objects associated with it might not resolve to the same value. References from within shared objects will normally be resolved by the dynamic linker to the virtual address of the function itself. References from within the executable file to a function defined in a shared object will normally be resolved by the link editor to the address of the procedure linkage table entry for that function within the executable file.

To allow comparisons of function addresses to work as expected, if an executable file references a function defined in a shared object, the link editor will place the address of the procedure linkage table entry for that function in its associated symbol table entry. [See "Symbol Values" in Chapter 4]. The dynamic linker

PROGRAM LOADING AND DYNAMIC LINKING

treats such symbol table entries specially. If the dynamic linker is searching for a symbol, and encounters a symbol table entry for that symbol in the executable file, it normally follows the rules below.

- If the st_shndx member of the symbol table entry is not SHN_UNDEF, the dynamic linker has found a definition for the symbol and uses its st_value member as the symbol's address.
- 2. If the st_shndx member is SHN_UNDEF and the symbol is of type STT_FUNC and the st_value member is not zero, the dynamic linker recognizes this entry as special and uses the st_value member as the symbol's address.
- 3. Otherwise, the dynamic linker considers the symbol to be undefined within the executable file and continues processing.

Some relocations are associated with procedure linkage table entries. These entries are used for direct function calls rather than for references to function addresses. These relocations are not treated in the special way described above because the dynamic linker must not redirect procedure linkage table entries to point to themselves.

Procedure Linkage Table

Much as the global offset table redirects position-independent address calculations to absolute locations, the procedure linkage table redirects position-independent function calls to absolute locations. The link editor cannot resolve execution transfers (such as function calls) from one executable or shared object to another. Consequently, the link editor arranges to have the program transfer control to entries in the procedure linkage table. On the Intel386 architecture, procedure linkage tables reside in shared text, but they use addresses in the private global offset table. The dynamic linker determines the destinations' absolute addresses and modifies the global offset table's memory image accordingly. The dynamic linker thus can redirect the entries without compromising the position-independence and sharability of the program's text. Executable files and shared object files have separate procedure linkage tables.

Dynamic Linking 5-7

Figure 5-6: Absolute Procedure Linkage Table

```
.PLT0: pushl got\_plus\_4
              *got_plus_8
       jmp
      nop; nop
      nop; nop
.PLT1: jmp
              *name1_in_GOT
      pushl $offset
       jmp
              .PLT0@PC
              *name2_in_GOT
.PLT2: jmp
      pushl $offset
              .PLT0@PC
       jmp
       . . .
```

Figure 5-7: Position-Independent Procedure Linkage Table

```
.PLT0: pushl 4(%ebx)
    jmp *8(%ebx)
    nop; nop
    nop; nop
.PLT1: jmp *name1@GOT(%ebx)
    pushl $offset
    jmp .PLT0@PC
.PLT2: jmp *name2@GOT(%ebx)
    pushl $offset
    jmp .PLT0@PC
...
```

PROGRAM LOADING AND DYNAMIC LINKING

NOTE

As the figures show, the procedure linkage table instructions use different operand addressing modes for absolute code and for position-independent code. Nonetheless, their interfaces to the dynamic linker are the same.

Following the steps below, the dynamic linker and the program "cooperate" to resolve symbolic references through the procedure linkage table and the global offset table.

- 1. When first creating the memory image of the program, the dynamic linker sets the second and the third entries in the global offset table to special values. Steps below explain more about these values.
- 2. If the procedure linkage table is position-independent, the address of the global offset table must reside in %ebx. Each shared object file in the process image has its own procedure linkage table, and control transfers to a procedure linkage table entry only from within the same object file. Consequently, the calling function is responsible for setting the global offset table base register before calling the procedure linkage table entry.
- 3. For illustration, assume the program calls name1, which transfers control to the label .PLT1.
- 4. The first instruction jumps to the address in the global offset table entry for name1. Initially, the global offset table holds the address of the following push1 instruction, not the real address of name1.
- 5. Consequently, the program pushes a relocation offset (*offset*) on the stack. The relocation offset is a 32-bit, non-negative byte offset into the relocation table. The designated relocation entry will have type R_386_JMP_SLOT, and its offset will specify the global offset table entry used in the previous jmp instruction. The relocation entry also contains a symbol table index, thus telling the dynamic linker what symbol is being referenced, name1 in this case.
- 6. After pushing the relocation offset, the program then jumps to .PLTO, the first entry in the procedure linkage table. The pushl instruction places the value of the second global offset table entry (got_plus_4 or 4(%ebx)) on the stack, thus giving the dynamic linker one word of identifying information. The program then jumps to the address in the third global offset table entry (got plus 8 or 8(%ebx)), which transfers control to the dynamic linker.
- 7. When the dynamic linker receives control, it unwinds the stack, looks at the designated relocation entry, finds the symbol's value, stores the "real" address for name1 in its global offset table entry, and transfers control to the desired destination.

Dynamic Linking 5-9

8. Subsequent executions of the procedure linkage table entry will transfer directly to name1, without calling the dynamic linker a second time. That is, the jmp instruction at .PLT1 will transfer to name1, instead of "falling through" to the push1 instruction.

The LD_BIND_NOW environment variable can change dynamic linking behavior. If its value is non-null, the dynamic linker evaluates procedure linkage table entries before transferring control to the program. That is, the dynamic linker processes relocation entries of type R_386_JMP_SLOT during process initialization. Otherwise, the dynamic linker evaluates procedure linkage table entries lazily, delaying symbol resolution and relocation until the first execution of a table entry.

NOTE

Lazy binding generally improves overall application performance, because unused symbols do not incur the dynamic linking overhead. Nevertheless, two situations make lazy binding undesirable for some applications. First, the initial reference to a shared object function takes longer than subsequent calls, because the dynamic linker intercepts the call to resolve the symbol. Some applications cannot tolerate this unpredictability. Second, if an error occurs and the dynamic linker cannot resolve the symbol, the dynamic linker will terminate the program. Under lazy binding, this might occur at arbitrary times. Once again, some applications cannot tolerate this unpredictability. By turning off lazy binding, the dynamic linker forces the failure to occur during process initialization, before the application receives control.

Program Interpreter

G

There is one valid program interpreter for programs conforming to the Intel386 ABI:

G

/usr/lib/libc.so.1

G

PROGRAM LOADING AND DYNAMIC LINKING

6 LIBRARIES

Shared Library Names	6-1
C Library	6-2
Additional Entry Points	6-2
Support Routines	6-3
Global Data Symbols	6-4
Application Constraints	6-4
System Data Interfaces	6-5
Data Definitions	6-5
■ Reentrancy Considerations	6-5
X Window Data Definitions	6-127
Motif 1.2 Data Definitions	6-192
TCP/IP Data Definitions	6-269

Table of Contents i

Shared Library Names		
The version number of below.	f the libraries named in the System V Generic ABI	is specified M M
Figure 6-1: Shared Lib	rary Names	M
	Library Reference Name	M
	libc.so.1	M
	libthread.so.1	M
	libdl.so.1	M
	libnsl.so.1	M
	libX11.so.5.0	M
	libXt.so.5.0	M
	libXext.so.5.0	M
	libXm.so.1.2	M
	libMrm.so.1.2	M

Shared Library Names

C Library M

Additional Entry Points

The following routines are included in the **libc** library to provide entry points for the required source-level interface listed in the *System V ABI*. A description and syntax symmary for each function follows the table.

```
Figure 6-2: libc Additional Required Entry Points
```

```
_fxstat _lxstat _xmknod _xstat nuname
_nuname
```

```
int _fxstat(int, int, struct stat *);
```

The semantics of this function are identical to those of the fstat(BA_OS) function described in the *System V Interface Definition, Edition 4.* Its only difference is that it requires an extra first argument whose value must be 2.

```
int _lxstat(int, char *, struct stat *);
```

The semantics of this function are identical to those of the lstat(BA_OS) function described in the *System V Interface Definition, Edition 4*. Its only difference is that it requires an extra first argument whose value must be 2.

```
int nuname(struct utsname *);
```

The semantics and syntax of this function are identical to those of the uname(BA_OS) function described in the *System V Interface Definition, Edition 4.* The symbol _nuname is also available with the same semantics.

```
int _xmknod(int, char *, mode_t, dev_t);
```

The semantics of this function are identical to those of the mknod(BA_OS) function described in the *System V Interface Definition, Edition 4.* Its only difference is that it requires an extra first argument whose value must be 2.

```
int _xstat(int, char *, struct stat *);
```

The semantics of this function are identical to those of the stat(BA OS) function described in the *System V Interface*

6-2 LIBRARIES

Definition, Edition 4. Its only difference is that it requires an extra first argument whose value must be 2.

M

Support Routines

Besides operating system services, **libc** contains the following processor-specific support routines.

Figure 6-3: libc, Support Routines

_fpstart _ _fpstart sbrk _sbrk

char *sbrk(int incr);

This function adds *incr* bytes to the *break value* and changes the allocated space accordingly. *Incr* can be negative, in which case the amount of allocated space is decreased. The break value is the address of the first allocation beyond the end of the data segment. The amount of allocated space increases as the break value increases. Newly allocated space is set to zero. If, however, the same memory space is reallocated to the same process, its contents are undefined. Upon successful completion, <code>sbrk</code> returns the old break value. Otherwise, it returns <code>-1</code> and sets <code>errno</code> to indicate the error. The symbol <code>_sbrk</code> is also available with the same semantics.

void __fpstart(void);

This function calls _fpstart(), to initialize the floating-point environment.

void _fpstart(void);

This function initializes the floating-point execution environment. It sets _fp_hw to the appropriate value. It sets the rounding mode to "nearest." It also resets the Intel387 control word to the default state.

C Library 6-3

Global Data Symbols

The libc library requires that some globel external data objects be defined for the routines to work properly. In addition to the corresponding data symbols listed in the *System V ABI*, the following symbols must be provided in the system library on all ABI-conforming systems implemented with the Intel386 architecture. Declarations for the data objects listed below can be found in the Data Definitions section of this chapter or immediately following the table.

Figure 6-4: libc, Global External Data Symbols

```
__flt_rounds _fp_hw __huge_val
```

extern int _fp_hw;

This variable describes the floating-point hardware available. If the value is zero, no floating-point support is present. If the value is 1, the floating-point support is provided by an Intel387 software emulator. If the value is 2, an 80287 chip is available. If the value is 3, an Intel387 chip is available. System software sets the value appropriately, before transferring control to main.

Application Constraints

As described above, libc provides symbols for applications. In a few cases, however, an application is obliged to provide symbols for the library. In addition to the application-provided symbols listed in this section of the *System V ABI*, conforming applications on the Intel386 architecture are also required to provide the following symbols.

extern end;

This symbol refers neither to a routine nor to a location with interesting contents. Instead, its address must correspond to the beginning of a program's dynamic allocation area, called the heap. Typically, the heap begins immediately after the data segment of the program's executable file.

extern const int _lib_version;

This variable's value specifies the compilation and execution mode for the program. If the value is zero, the program wants to preserve the semantics of older (pre-ANSI) C, where conflicts exist with ANSI. Otherwise, the value is non-zero, and the program wants ANSI C semantics.

6-4 LIBRARIES

System Data Interfaces

Data Definitions

This section contains standard data definitions that describe system data. These files are referred to by their names in angle brackets: <name.h> and <sys/name.h>. Included in these data definitions are macro definitions and data definitions.

The data objects described in this section are part of the interface between an ABI-conforming application and the underlying ABI-conforming system where it will run. While an ABI-conforming system must provide these interfaces, it is not required to contain the actual data definitions referenced here. Programmers should observe that the sources of the structures defined in these data definitions are defined in SVID.

ANSI C serves as the ABI reference programming language, and data definitions are specificed in ANSI C format. The C language is used here as a convenient notation. Using a C language description of these data objects does *not* preclude their use by other programming languages.

M

M

Μ

M

Reentrancy Considerations

New conventions have been added to accomodate the new requirements of reentrancy. Some historic binary code sequences are inherently non-reentrant. Unless great care is taken, multi-threaded applications cannot safely use such sequences. The most portable (i.e. those guaranteed to work in all cases) are those that are marked as reentrant in this chapter. For the ABI, this sometimes requires that two definitions exist for these interfaces, one that is reentrant and one that is not. These are indicated by comments that define which of the alternate definitions is reentrant. These alternatives are not selected at run-time, but are intended to be bound at application build time.

M M M M

M

NOTE

All information presented in the figures marked with * are new to the Fourth Edition of the psABI.

M

System Data Interfaces

```
Figure 6-5: <aio.h>*
```

```
struct aiocb {
       int
                       aio_fildes;
       volatile void* aio_buf;
       size_t
                       aio_nbytes;
       off_t
                       aio_offset;
       int
                       aio_reqprio;
       struct sigevent aio_sigevent;
                       aio_lio_opcode;
       ssize_t
       int
       int
       void
       int
} ;
#define AIO_CANCELED
                        (0)
#define AIO_ALLDONE
                        (1)
#define AIO_NOTCANCELED (2)
#define LIO_NOWAIT
                        (0)
#define LIO_WAIT
                        (1)
#define LIO_NOP
                        (0)
#define LIO_READ
                        (1)
#define LIO_WRITE
                        (2)
```

Figure 6-6: <assert.h>

```
extern void __assert(const char *, const char *, int);
#define assert(EX) (void)((EX)||(__assert(#EX, __FILE__, __LINE__), 0))
```

6-6 LIBRARIES

Figure 6-7: <ctype.h>

```
#define U
             01
#define _L
             02
#define _N
             04
#define _S
             010
#define _P
             020
             040
#define C
#define _B
             0100
#define _X
             0200
extern unsigned char
                          __ctype[];
#define isalpha(c) ((__ctype+1)[c]&(_U|_L))
#define isupper(c) ((__ctype+1)[c]&_U)
#define islower(c) ((__ctype+1)[c]&_L)
#define isdigit(c) ((__ctype+1)[c]&_N)
#define isxdigit(c) ((__ctype+1)[c]&_X)
#define isalnum(c) ((__ctype+1)[c]&(_U|_L|_N))
#define isspace(c) ((__ctype+1)[c]&_S)
#define ispunct(c) ((__ctype+1)[c]&_P)
\#define isprint(c) ((\__ctype+1)[c]&(\_P|\_U|\_L|\_N|\_B))
\#define isgraph(c) ((\_ctype+1)[c]&(\_P|\_U|\_L|\_N))
#define iscntrl(c) ((__ctype+1)[c]&_C)
#define isascii(c) (!((c)&~0177))
#define _toupper(c) ((__ctype+258)[c])
#define _tolower(c) ((__ctype+258)[c])
#define toascii(c) ((c)&0177)
```

CAUTION

The data definitions in ctype.h are moved to Level 2 as of January 1, 1993. In order to correctly function in an internationalized environment, applications are encouraged to use the functions in libc instead.

System Data Interfaces

6-7

M

Figure 6-8: <dirent.h>

```
typedef struct {
      int
                    dd_fd;
      int
                    dd_loc;
      int
                    dd_size;
      char
                    *dd_buf;
} DIR;
struct dirent {
      ino_t
                    d_ino;
      off_t
                    d_off;
      unsigned short
                          d_reclen;
                    d_name[1];
};
```

Figure 6-9: <dlfcn.h>*

```
#define RTLD_LAZY 1
#define RTLD_NOW 2
#define RTLD_GLOBAL 4
```

6-8 LIBRARIES

Figure 6-10: <elf.h>*, Part 1 of 6

```
#define ELF32 FSZ ADDR
#define ELF32_FSZ_HALF
                          2
#define ELF32_FSZ_OFF
                          4
#define ELF32_FSZ_SWORD
                          4
#define ELF32_FSZ_WORD
#define EI_NIDENT
typedef struct {
      unsigned chare_ident[EI_NIDENT];
      Elf32_Half e_type;
      Elf32 Half e machine;
                   e_version;
      Elf32_Word
      Elf32_Addr
                  e_entry;
      Elf32_Off
                   e_phoff;
      Elf32_Off
                   e_shoff;
      Elf32_Word
                   e_flags;
      Elf32_Half
                   e_ehsize;
      Elf32_Half
                   e_phentsize;
      Elf32_Half
                   e_phnum;
      Elf32_Half
                   e_shentsize;
      Elf32_Half
                   e_shnum;
      Elf32 Half
                   e_shstrndx;
} Elf32_Ehdr;
#define ELFMAG0
                          0x7f
#define ELFMAG1
                          'E'
#define ELFMAG2
                          'L'
#define ELFMAG3
                          'F'
#define ELFMAG
                          "\177ELF"
#define SELFMAG
```

System Data Interfaces

Figure 6-11: <elf.h>*, Part 2 of 6

	#define	EI_MAG0	0
١	#define	EI_MAG1	1
١	#define	EI_MAG2	2
١	#define	EI_MAG3	3
١		EI_CLASS	4
١		EI_DATA	5
١		EI_VERSION	6
١	#define	EI_PAD	7
١			
١	#define	ELFCLASSNONE	0
١		ELFCLASS32	1
١		ELFCLASS64	2
١		ELFCLASSNUM	3
١	**	ELFDATANONE	0
١		ELFDATA2LSB	1
١		ELFDATA2MSB	2
١	#define	ELFDATANUM	3
١			
١		ET_NONE	0
١	#define		1
١		ET_EXEC	2
١	#define		3
١		ET_CORE	4 5
١	#define	E.IINOM	5
	#define	ET_LOPROC	0xff00
١	#define		0xffff
١	пасттіс	H1_IIII100	OMILLI
١	#define	EM NONE	0
١	#define	_	1
١	#define	EM_SPARC	2
١	#define	EM_386	3
	#define	EM_68K	4
	#define	EM_88K	5
	#define	EM_486	6
	#define	EM_860	7
	#define	EM_NUM	8

6-10 LIBRARIES

Figure 6-12: <elf.h>*, Part 3 of 6

```
#define EV_NONE
#define EV_CURRENT
                          1
#define EV_NUM
typedef struct {
      Elf32_Word
                   p_type;
      Elf32_Off
                   p_offset;
      Elf32_Addr
                   p_vaddr;
      Elf32_Addr
                   p_paddr;
      Elf32_Word
                   p_filesz;
      Elf32_Word
                   p_memsz;
      Elf32_Word
                   p_flags;
      Elf32_Word
                   p_align;
} Elf32_Phdr;
#define PT_NULL
                          0
#define PT_LOAD
                          1
#define PT_DYNAMIC
                          2
#define PT_INTERP
                          3
#define PT_NOTE
                          4
                          5
#define PT SHLIB
#define PT_PHDR
                          6
#define PT_NUM
                          7
#define PT_LOPROC
                    0x70000000
#define PT_HIPROC
                   0x7fffffff
#define PF_R
                    0x4
#define PF_W
                    0x2
#define PF_X
                    0x1
#define PF_MASKPROC 0xf0000000
```

System Data Interfaces

Figure 6-13: <elf.h>*, Part 4 of 6

```
typedef struct {
      Elf32_Word
                    sh_name;
      Elf32_Word
                    sh_type;
      Elf32_Word
                    sh_flags;
      Elf32_Addr
                    sh_addr;
      Elf32_Off
                    sh_offset;
      Elf32_Word
                    sh size;
                    sh_link;
      Elf32_Word
      Elf32_Word
                    sh_info;
      Elf32_Word
                    sh_addralign;
      Elf32_Word
                    sh_entsize;
} Elf32_Shdr;
#define SHT_NULL
                           0
#define SHT_PROGBITS
                          1
#define SHT_SYMTAB
#define SHT_STRTAB
                           3
#define SHT_RELA
                           4
#define SHT_HASH
                           5
#define SHT_DYNAMIC
                           6
                          7
#define SHT_NOTE
#define SHT_NOBITS
                           8
#define SHT REL
                           9
#define SHT_SHLIB
                          10
#define SHT_DYNSYM
                           11
#define SHT_NUM
#define SHT_LOUSER 0x80000000
#define SHT_HIUSER 0xffffffff
#define
             SHT_LOPROC
                          0x70000000
#define
             SHT_HIPROC
                           0x7fffffff
#define SHF_MASKPROC
                           0xf0000000
#define SHF WRITE
                           0x1
#define SHF_ALLOC
                           0x2
#define SHF_EXECINSTR
                           0x4
```

6-12 LIBRARIES

Figure 6-14: <elf.h>*, Part 5 of 6

```
#define SHN UNDEF
                           0
#define SHN_LORESERVE
                           0xff00
#define SHN_ABS
                           0xfff1
#define SHN_COMMON
                           0xfff2
#define SHN_HIRESERVE
                          0xffff
#define SHN_LOPROC
                          0xff00
#define SHN_HIPROC
                          0xff1f
typedef struct {
      Elf32_Word
                          st_name;
      Elf32_Addr
                          st_value;
      Elf32 Word
                          st size;
      unsigned char
                          st_info;
      unsigned char
                          st_other;
      Elf32_Half
                          st_shndx;
} Elf32_Sym;
#define STN_UNDEF
#define ELF32_ST_BIND(info)
                                        ((info) >> 4)
#define ELF32_ST_TYPE(info)
                                        ((info) & 0xf)
\#define ELF32\_ST_INFO(bind,type) (((bind)<<4)+((type)&0xf))
#define STB_LOCAL
#define STB_GLOBAL 1
#define STB_WEAK
#define STB_NUM
#define STB_LOPROC 13
#define STB_HIPROC 15
```

System Data Interfaces

```
Figure 6-15: <elf.h>*, Part 6 of 6
```

```
#define STT NOTYPE 0
#define STT_OBJECT 1
#define STT_FUNC
#define STT_SECTION 3
#define STT_FILE
#define STT_NUM
                   5
#define STT_LOPROC 13
#define STT_HIPROC 15
typedef struct {
      Elf32_Addr r_offset;
      Elf32_Word r_info;
} Elf32_Rel;
typedef struct {
      Elf32_Addr
                 r_offset;
      Elf32_Word r_info;
      Elf32_Sword r_addend;
} Elf32_Rela;
#define ELF32_R_SYM(info) ((info)>>8)
#define ELF32_R_TYPE(info)((unsigned char)(info))
#define ELF32_R_INFO(sym,type) (((sym)<<8)+(unsigned char)(type))</pre>
```

6-14 LIBRARIES

Figure 6-16: <errno.h>, Part 1 of 3

#define	EPERM	1
#define	ENOENT	2
#define	ESRCH	3
#define	EINTR	4
#define	EIO	5
#define	ENXIO	6
#define	E2BIG	7
#define	ENOEXEC	8
#define	EBADF	9
#define	ECHILD	10
#define	EAGAIN	11
#define	ENOMEM	12
#define	EACCES	13
#define	EFAULT	14
#define	ENOTBLK	15
#define	EBUSY	16
#define	EEXIST	17
#define	EXDEV	18
#define	ENODEV	19
#define	ENOTDIR	20
#define	EISDIR	21
#define	EINVAL	22
#define	ENFILE	23
#define	EMFILE	24
#define	ENOTTY	25
#define	ETXTBSY	26
#define	EFBIG	27

System Data Interfaces

Figure 6-17: <errno.h>, Part 2 of 3

#define	ENOSPC	28
#define	ESPIPE	29
#define	EROFS	30
#define	EMLINK	31
#define	EPIPE	32
#define	EDOM	33
#define	ERANGE	34
#define	ENOMSG	35
#define	EIDRM	36
#define	ECHRNG	37
#define	EL2NSYNC	38
#define	EL3HLT	39
#define	EL3RST	40
#define	ELNRNG	41
#define	EUNATCH	42
#define	ENOCSI	43
#define	EL2HLT	44
#define	EDEADLK	45
#define	ENOLCK	46
#define	ENOSTR	60
#define	ENODATA	61
#define	ETIME	62
#define	ENOSR	63
#define	ENONET	64
#define	ENOPKG	65
#define	EREMOTE	66
#define	ENOLINK	67

6-16 LIBRARIES

Figure 6-18: <errno.h>, Part 3 of 3

#define EADV	68	
#define ESRMNT	69	
#define ECOMM	70	
#define EPROTO	71	
#define EMULTIHOP	74	
#define EBADMSG	77	
#define ENAMETOOLONG	78	
#define EOVERFLOW	79	
#define ENOTUNIQ	80	
#define EBADFD	81	
#define EREMCHG	82	
#define ENOSYS	89	
#define ELOOP	90	
#define ERESTART	91	
#define ESTRPIPE	92	
#define ENOTEMPTY	93	
#define EUSERS	94	
#define ECONNABORTED	130	
#define CONNRESET	131	
#define ECONNREFUSED		146
#define EINPROGRESS	150	
#define ESTALE	151	
#define ECANCELED	158	
/* Non-reentrant */		
extern int errno;		
/* Reentrant */		
#define errno (*thr	_errno())	

Figure 6-19: <fcntl.h>, Part 1 of 2

#define	O_RDONLY	0
#define	O_WRONLY	1
#define	O_RDWR	2
#define	O_NDELAY	0x04
#define	O_APPEND	0x08
#define	O_SYNC	0x10
#define	O_NONBLOCK	0x80
#define	O_CREAT	0x100
#define	O_TRUNC	0x200
#define	O_EXCL	0x400
#define	O_NOCTTY	0x800
#define	F_DUPFD	0
#define	F_GETFD	1
#define	F_SETFD	2
#define	F_GETFL	3
#define	F_SETFL	4
#define	F_GETLK	14
#define	F_SETLK	6
#define	F_SETLKW	7

6-18 LIBRARIES

Figure 6-20: <fcntl.h>, Part 2 of 2

```
typedef struct flock {
      short l_type;
      short l_whence;
      off_t l_start;
      off_t l_len;
      long l_sysid;
      pid_t l_pid;
      long pad[4];
} flock_t;
#define F_RDLCK
                          01
#define F_WRLCK
                          02
#define F_UNLCK
                          03
#define O_ACCMODE
                          3
#define FD_CLOEXEC
                          1
```

Figure 6-21: <float.h>, Single-Precision

```
extern int
                    __flt_rounds;
                          __flt_rounds
#define FLT_ROUNDS
#define FLT_RADIX
                          2
#define FLT_MANT_DIG
#define FLT_EPSILON
                          1.19209290E-07F
#define FLT_DIG
#define FLT_MIN_EXP
                          (-125)
#define FLT_MIN
                          1.17549435E-38F
#define FLT_MIN_10_EXP
                          (-37)
#define FLT_MAX_EXP
                          (+128)
#define FLT MAX
                          3.40282347E+38F
#define FLT_MAX_10_EXP
                          (+38)
```

Figure 6-22: <float.h>, Double-Precision

```
#define DBL_MANT_DIG
                          53
#define DBL_EPSILON
                          2.2204460492503131E-16
#define DBL DIG
                          15
#define DBL_MIN_EXP
                          (-1021)
                          2.2250738585072014E-308
#define DBL_MIN
#define DBL_MIN_10_EXP
                          (-307)
#define DBL_MAX_EXP
                          (+1024)
#define DBL MAX
                          1.7976931348623157E+308
#define DBL_MAX_10_EXP
                          (+308)
```

6-20 LIBRARIES

Figure 6-23: <float.h>, Extended-Precision

#define LDBL_MANT_DIG	64	G
#define LDBL_EPSILON	1.084202172485504434e-19	G
#define LDBL_DIG	18	G
#define LDBL_MIN_EXP	-16381	G
#define LDBL_MIN	3.362103143112093506e-4932	G
#define LDBL_MIN_10_EXP	-4931	G
#define LDBL_MAX_EXP	16384	G
#define LDBL_MAX	1.189731495347231765e+4932	G
#define LDBL_MAX_10_EXP	4932	G
		1

Figure 6-24: <fmtmsg.h>, Part 1 of 2

#define	MM_NULL	0L
#define #define #define #define #define #define #define #define	MM_HARD MM_SOFT MM_FIRM MM_RECOVER MM_NRECOV MM_APPL MM_UTIL MM_OPSYS MM_PRINT	0x00000001L 0x00000002L 0x00000004L 0x00000100L 0x00000200L 0x00000008L 0x00000010L 0x00000020L 0x00000020L
	MM_CONSOLE	0x00000040L

Figure 6-25: <fmtmsg.h>, Part 2 of 2

```
#define MM NOSEV
                          0
#define MM_HALT
                          1
#define MM_ERROR
                          2
#define MM_WARNING
                          3
#define MM_INFO
#define MM_NULLLBL
                          ((char *) NULL)
#define MM_NULLSEV
                          MM_NOSEV
#define MM_NULLMC
                          MM_NULL
#define MM_NULLTXT
                           ((char *) NULL)
#define MM_NULLACT
                          ((char *) NULL)
#define MM_NULLTAG
                          ((char *) NULL)
#define MM_NOTOK
#define MM_OK
                          0x00
#define MM_NOMSG
                          0x01
#define MM_NOCON
                          0x04
```

Figure 6-26: <fnmatch.h>*

#define	FNM_PATHNAME	0x001
#define	FNM_PERIOD	0x002
#define	FNM_NOESCAPE	0x004
#define	FNM_BADRANGE	800x0
#define	FNM_EXTENDED	0x020
#define	FNM_NOSYS	(-1)
#define	FNM_NOMATCH	(-2)

6-22 LIBRARIES

Figure 6-27: <ftw.h>

```
#define FTW_F
                           0
#define FTW_D
                           1
                           2
#define FTW_DNR
#define FTW_NS
                           3
#define FTW_SL
#define FTW_DP
                           6
#define FTW_SLN
#define FTW_PHYS
                           01
                           02
#define FTW_MOUNT
#define FTW_CHDIR
                           04
                           010
#define FTW_DEPTH
struct FTW {
      int
             quit;
      int
             base;
      int
             level;
};
#define FTW_SKD
                           1
#define FTW_FOLLOW
                           2
#define FTW_PRUNE
                           4
```

Figure 6-28: <glob.h>*

```
#define GLOB APPEND
                           0 \times 0001
#define GLOB_DOOFFS
                           0x0002
#define GLOB_ERR
                           0 \times 0004
#define GLOB_MARK
                           0x0008
#define GLOB_NOCHECK
                           0x0010
#define GLOB_NOSORT
                           0x0020
#define GLOB_NOESCAPE
                           0x0040
#define GLOB_OKAYDOT
                           0x0200
#define GLOB_BADRANGE
                           0x0400
#define GLOB_EXTENDED
                           0x1000
#define GLOB NOSYS
                            (-1)
#define GLOB_ABORTED
                            (-2)
#define GLOB_NOSPACE
                            (-3)
#define GLOB_NOMATCH
                            (-4)
typedef struct
      void
                    *;
      char
                    **gl_pathv;
      size_t
                    gl_pathc;
                    gl_offs;
      size_t
} glob_t;
```

6-24 LIBRARIES

Figure 6-29: <grp.h>

```
struct group {
    char *gr_name;
    char *gr_passwd;
    gid_t gr_gid;
    char **gr_mem;
};
```

Figure 6-30: <iconv.h>*

```
typedef void *iconv_t;
```

System Data Interfaces

Figure 6-31: <sys/ipc.h>

```
struct ipc_perm {
      uid_t
                   uid;
      gid_t
                   gid;
      uid_t
                   cuid;
      gid_t
                   cgid;
                   mode;
      mode_t
      ulong
                    seq;
                   key;
      key_t
      long
                   pad[4];
};
#define IPC_CREAT
                    0001000
#define IPC_EXCL
                    0002000
#define IPC_NOWAIT 0004000
#define IPC_PRIVATE (key_t)0
#define IPC_RMID
                          10
#define IPC_SET
                          11
#define IPC_STAT
                          12
```

6-26 LIBRARIES

Figure 6-32: <langinfo.h>, Part 1 of 2

#define	DAY_1	1
#define	DAY_2	2
#define	DAY_3	3
#define	DAY_4	4
#define	DAY_5	5
#define	DAY_6	6
#define	DAY_7	7
#define	ABDAY_1	8
#define	ABDAY_2	9
#define	ABDAY_3	10
#define	ABDAY_4	11
#define	ABDAY_5	12
#define	ABDAY_6	13
#define	ABDAY_7	14
#define	MON_1	15
#define	MON_2	16
#define		17
#define		18
#define	_	19
#define	MON_6	20
#define	MON_7	21
#define	MON_8	22
#define		23
#define	·	24
#define		25
#define	MON_12	26

Figure 6-33: <langinfo.h>, Part 2 of 2

-			
	#define	ABMON_1	27
	#define	ABMON_2	28
	#define	ABMON_3	29
	#define	ABMON_4	30
	#define	ABMON_5	31
	#define	ABMON_6	32
	#define	ABMON_7	33
	#define	ABMON_8	34
	#define	ABMON_9	35
	#define	ABMON_10	36
	#define	ABMON_11	37
	#define	ABMON_12	38
	#define	RADIXCHAR	39
	#define	THOUSEP	40
	#define	YESSTR	41
	#define	NOSTR	42
	#define	CRNCYSTR	43
	#define	D_T_FMT	44
	#define	D_FMT	45
	#define	T_FMT	46
	#define	AM_STR	47
	#define	PM_STR	48
	#define	CODESET	49
	#define	T_FMT_AMPM	50
	#define	ERA	51
		ERA_D_FMT	52
	#define	ERA_D_T_FMT	53
		ERA_T_FMT	54
	#define	ALT_DIGITS	55
		YESEXPR	56
	#define	NOEXPR	57

6-28 LIBRARIES

Figure 6-34: simits.h>, Part 1 of 2

```
#define CHAR BIT
                          8
#define SCHAR_MIN
                          (-128)
#define SCHAR MAX
                          127
                          255
#define UCHAR_MAX
#define MB_LEN_MAX
#define CHAR MIN
                          SCHAR MIN
#define CHAR_MAX
                          SCHAR_MAX
#define SHRT_MIN
                          (-32768)
#define SHRT_MAX
                          32767
#define USHRT MAX
                         65535
#define INT_MIN
                          (-2147483647-1)
#define INT_MAX
                          2147483647
#define UINT_MAX
                          4294967295
#define LONG_MIN
                          (-2147483647-1)
#define LONG_MAX
                          2147483647
#define ULONG_MAX
                          4294967295
#define ARG_MAX
#define LINK_MAX
#define MAX_CANON
#define MAX INPUT
#define NGROUPS_MAX
#define PATH MAX
#define PIPE_BUF
#define TMP_MAX
#define PASS_MAX
#define CHILD MAX
/* starred values vary and should be
      retrieved using sysconf() or pathconf() */
```

Figure 6-35: imits.h>, Part 2 of 2

#define NL_	_ARGMAX	9
#define NL_	_LANGMAX	14
#define NL_	_MSGMAX	32767
#define NL_	_NMAX	1
#define NL_	_SETMAX	255
#define NL_	_TEXTMAX	255
#define NZE	ERO	20
#define WOF	RD_BIT	32
#define LON	NG_BIT	32
#define DBI	L_DIG	15
#define DBI	L_MAX	1.7976931348623157E+308
#define DBI	L_MIN	2.2250738585072014E-308
#define FL7	Γ_DIG	6
#define FL7	Γ_MAX	3.40282347E+38F
#define FL7	 ΓMIN	1.17549435E-38F
	_	
#define FCF	HR MAX	1048576
	_	

6-30 LIBRARIES

Figure 6-36: <locale.h>

```
struct lconv {
      char
             *decimal_point;
      char
             *thousands_sep;
      char
             *grouping;
      char
             *int_curr_symbol;
             *currency_symbol;
      char
      char
             *mon_decimal_point;
      char
             *mon_thousands_sep;
      char
             *mon_grouping;
      char
             *positive_sign;
      char *negative_sign;
      char int_frac_digits;
      char frac_digits;
      char p_cs_precedes;
      char p_sep_by_space;
      char n_cs_precedes;
      char n_sep_by_space;
      char
             p_sign_posn;
      char n_sign_posn;
};
#define LC_CTYPE
                          0
#define LC NUMERIC
                          1
#define LC_TIME
                          2
#define LC_COLLATE
                          3
#define LC_MONETARY
                          4
#define LC_MESSAGES
                          5
#define LC_ALL
                          6
```

```
Figure 6-37: <lwpsynch.h>*
```

```
typedef volatile struct {
    char wanted;
    _simplelock_t lock;
} lwp_mutex_t;

typedef volatile struct {
    char wanted;
} lwp_cond_t;
```

Figure 6-38: <machlock.h>*

 ${\tt typedef\ volatile\ unsigned\ char\ _simplelock_t;}$

Figure 6-39: <math.h>

extern const double __huge_val; G
#define HUGE_VAL __huge_val G

6-32 LIBRARIES

Figure 6-40: <sys/mman.h>

				-
#define PROT	_READ	0x1		
#define PROT	_WRITE	0x2		
#define PROT	r_exec	0x4		
#define PROT	I_NONE	0x0		
#define MAP_	_	1		
#define MAP_	_PRIVATE	2		
#define MAP_	_FIXED	0x10		
W 3 C' 350 C	~-~	0 0		
#define MS_S		0x0		
#define MS_A		0x1		
#define MS_1	INVALIDATE	0x2		
#dofina DBOO	C TEXT (PROT	evec l	DDOT DEAD)	G
		_	PROT_WRITE PROT_EXEC)	G
#deline Proc	DAIA (PROI_	_KEAD	PROI_WRITE PROI_EXEC)	G
#define SHAF	RED	0x10		G
#define PRIV	/ATE	0x20		G
#define	MC_SYNC		1	G
#define	MC_LOCK		2	G
#define	MC_UNLOCK	3		G
#define	MC_LOCKAS	5		G
#define	MC_UNLOCKAS	6		G
#define	MCL_CURRENT			G
#define	MCL_FUTURE	0x2		G

Figure 6-41: <sys/mod.h>*

```
#define VOID
                          void
#define MAXPATHLEN
                          1024
#define MODMAXLINKINFOLEN 32
struct modspecific_stat {
      char mss_linkinfo[MODMAXLINKINFOLEN];
      int
           mss_type;
           mss_p0[2];
      int
            mss_p1[2];
      int
};
#define MODMAXLINK
struct modstatus {
      int
                          ms_id;
      VOID
                          *ms_base;
                          ms_size;
      unsigned int
      int
                          ms_rev;
      char
                          ms_path[MAXPATHLEN];
      time_t
                          ms_unload_delay;
      int
                          ms_refcnt;
                          ms_depcnt;
      int
      struct modspecific_stat ms_msinfo[MODMAXLINK];
};
```

6-34 LIBRARIES

Figure 6-42: <sys/mount.h>

#define #define	MS_RDONLY MS_FSS MS_DATA MS_HADBAD	0x01 0x02 0x04 0x08
#define	MS_NOSUID MS_REMOUNT MS_NOTRUNC	0x10 0x20 0x40

Figure 6-43: <sys/msg.h>

```
#define MSG_NOERROR
                           010000
struct msqid_ds {
      struct ipc_perm
                           msg_perm;
      struct msg
                    *msg_first;
                    *msg_last;
      struct msg
      ulong
                    msg_cbytes;
      ulong
                    msg_qnum;
      ulong
                    msg_qbytes;
      pid_t
                    msg_lspid;
                    msg_lrpid;
      pid_t
      time_t
                    msg_stime;
      long
                    msg_pad1;
      time_t
                    msg_rtime;
      long
                    msg_pad2;
      time_t
                    msg_ctime;
                    msg_pad3;
      long
      long
                    msg_pad4[4];
};
struct msg {
      struct msg
                    *msg_next;
      long
                    msg_type;
      ushort
                    msg_ts;
      short
                    msg_spot;
};
```

6-36 LIBRARIES

Figure 6-44: <netconfig.h>, Part 1 of 2

struct netconfig { char *nc_netid; unsigned long nc_semantics; unsigned long nc_flag; char *nc_protofmly; char *nc_proto; char *nc_device; unsigned long nc_nlookups; char **nc_lookups; unsigned long nc_unused[8]; }; #define NC_TPI_CLTS 1 2 #define NC_TPI_COTS #define NC_TPI_COTS_ORD 3 #define NC_TPI_RAW 4 #define NC_NOFLAG 00 01 #define NC_VISIBLE

System Data Interfaces

6-37

G

Figure 6-45: <netconfig.h>, Part 2 of 2

#define	NC_NOPROTOFMLY	" _ "
#define	NC_LOOPBACK	"loopback"
#define	NC_INET	"inet"
#define	NC_IMPLINK	"implink"
#define	NC_PUP	"pup"
#define	NC_CHAOS	"chaos"
#define	NC_NS	"ns"
#define	NC_NBS	"nbs"
#define	NC_ECMA	"ecma"
#define	NC_DATAKIT	"datakit"
#define	NC_CCITT	"ccitt"
#define	NC_SNA	"sna"
#define	NC_DECNET	"decnet"
#define	NC_DLI	"dli"
#define	NC_LAT	"lat"
#define	NC_HYLINK	"hylink"
#define	NC_APPLETALK	"appletalk"
#define	NC_NIT	"nit"
#define	NC_IEEE802	"ieee802"
#define	NC_OSI	"osi"
#define	NC_X25	"x25"
#define	NC_OSINET	"osinet"
#define	NC_GOSIP	"gosip"
#define	NC_NETWARE	"netware"
#define	NC_NOPROTO	" _ "
#define	NC_TCP	"tcp"
#define	NC_UDP	"udp"
#define	NC_ICMP	"icmp"
#define	NC_IPX	"ipx"
#define	NC_SPX	"spx"

6-38 LIBRARIES

Figure 6-46: <netdir.h>, Part 1 of 2

```
struct nd_addrlist {
      int
                          n_cnt;
      struct netbuf
                           *n addrs;
};
struct nd_hostservlist {
      int
                          h_cnt;
      struct nd_hostserv *h_hostservs;
};
struct nd_hostserv {
      char
                    *h host;
      char
                    *h_serv;
};
#define ND_HOSTSERV
#define ND_HOSTSERVLIST
                                 1
#define ND_ADDR
#define ND_ADDRLIST
                                 3
                                 -2
#define ND_BADARG
#define ND_NOMEM
                                 -1
#define ND OK
                                 0
#define ND_NOHOST
                                 1
#define ND_NOSERV
                                 2
#define ND_NOSYM
                                 3
#define ND_OPEN
                                 4
                                 5
#define ND_ACCESS
#define ND_UKNWN
                                 6
                                 7
#define ND_NOCTRL
#define ND_FAILCTRL
                                 8
#define ND_SYSTEM
                                 9
#define ND_NOERRMEM
                                 10
#define ND NOLIB
                                 11
#define ND_XTIERROR
                                 12
#define ND_BADSTATE
                                 13
```

6-39

M

M

M

M

Figure 6-47: <netdir.h>, Part 2 of 2

#define	ND_SET_BROADCAST	1
#define	ND_SET_RESERVEDPORT	2
#define	ND_CHECK_RESERVEDPORT	3
#define	ND_MERGEADDR	4
#define	ND_CLEAR_BROADCAST	5
#define	ND_SET_REUSEADDR	6
#define	ND_CLEAR_REUSEADDR	7
#define	HOST_SELF	"\\1"
#define	HOST_ANY	"\\2"
#define	HOST_BROADCAST	"\\3"

Figure 6-48: <nl_types.h>

```
#define NL_SETD 1

typedef int nl_item;
typedef void *nl_catd;
```

6-40 LIBRARIES

Figure 6-49: <sys/param.h>

#define #define #define		256 100 10000000
#define	NGROUPS_UMIN	0
#define	NBPSCTR	512
#define	MAXPATHLEN MAXSYMLINKS MAXNAMELEN	1024 20 256
#define	NADDR	13
#define	PIPE_MAX	5120
#define	NBBY	8
#define	MAXFRAG	8

Figure 6-50: <poll.h>

```
struct pollfd {
      int
             fd;
      short events;
      short revents;
};
#define POLLIN
                           0x0001
#define POLLPRI
                           0x0002
#define POLLOUT
                           0x0004
#define POLLRDNORM
                           0x0040
#define POLLWRNORM
                          POLLOUT
#define POLLRDBAND
                          0x0080
#define POLLWRBAND
                          0x0100
#define POLLNORM
                          POLLRDNORM
#define POLLERR
                           0x0008
#define POLLHUP
                           0x0010
#define POLLNVAL
                           0x0020
```

6-42 LIBRARIES

Figure 6-51: <sys/priocntl.h>*

```
#define PC_GETCID
#define PC_GETCLINFO
                                1
#define PC_SETPARMS
#define PC_GETPARMS
                          3
#define PC_CLNULL
                          -1
#define PC_CLNMSZ
                  16
#define PC_CLINFOSZ (32 / sizeof(long))
#define PC_CLPARMSZ (32 / sizeof(long))
typedef struct pcinfo {
      id_t pc_cid;
      char pc_clname[PC_CLNMSZ];
      long pc_clinfo[PC_CLINFOSZ];
} pcinfo_t;
typedef struct pcparms {
      id_t pc_cid;
      long pc_clparms[PC_CLPARMSZ];
} pcparms_t;
```

Figure 6-52: <sys/procset.h>

```
#define P_INITPID
                           1
#define P_INITUID
                           0
#define P_INITPGID
typedef enum idtype {
      P_PID,
      P_PPID,
      P_PGID,
      P_SID,
      P_CID,
      P_UID,
      P_GID,
      P_ALL
} idtype_t;
typedef enum idop {
      POP_DIFF,
      POP_AND,
      POP_OR,
      POP_XOR
} idop_t;
typedef struct procset {
      idop_t
                    p_op;
      idtype_t
                    p_lidtype;
      id_t
                    p_lid;
                    p_ridtype;
      idtype_t
      id_t
                    p_rid;
} procset_t;
```

6-44 LIBRARIES

Figure 6-53: <pwd.h>

```
struct passwd {
      char
                    *pw_name;
      char
                    *pw_passwd;
      uid_t
                    pw_uid;
      gid_t
                    pw_gid;
      char
                    *pw_age;
                    *pw_comment;
      char
      char
                    *pw_gecos;
                    *pw_dir;
      char
                    *pw_shell;
      char
};
```

Figure 6-54: <regex.h>*, Part 1 of 2

```
#define REG NOTBOL
                            0x000001
#define REG_NOTEOL
                            0 \times 000002
#define REG_NONEMPTY
                            0 \times 000004
#define REG_OR
                            0x000001
#define REG_PLUS
                            0 \times 000002
#define REG_QUEST
                            0 \times 000004
#define REG_BRACES
                            0x000008
#define REG_PARENS
                            0x000010
#define REG_ANCHORS
                            0x000020
#define REG_NOBACKREF
                            0 \times 000040
#define REG_NOAUTOQUOTE
                            0x000080
#define REG EXTENDED
                            (REG_OR | REG_PLUS | REG_QUEST |
              REG_BRACES | REG_PARENS | REG_ANCHORS |
              REG_NOBACKREF | REG_NOAUTOQUOTE)
#define REG ICASE
                            0x000100
#define REG_NOSUB
                            0 \times 000200
#define REG_NEWLINE
                            0x000400
#define REG_ONESUB
                            0x000800
#define REG_BADRANGE
                            0x004000
#define REG_ANGLES
                            0x040000
#define REG ESCNL
                            0x080000
#define REG_OLDBRE (REG_BADRANGE | REG_ANGLES | REG_ESCNL)
```

6-46 LIBRARIES

Figure 6-55: <regex.h>*, Part 2 of 2

```
#define REG ENOSYS
                          (-1)
#define REG_NOMATCH
                          1
#define REG_BADPAT
                          2
#define REG_ECOLLATE
                          3
#define REG_ECTYPE
#define REG_EESCAPE
                          7
#define REG_ESUBREG
                          8
                          9
#define REG_EBRACK
#define REG_NOPAT
                          12
#define REG_EPAREN
                          13
#define REG_EBRACE
                          14
#define REG BADBR
                          15
#define REG_ERANGE
                          16
#define REG_ESPACE
                          17
#define REG_BADRPT
                          18
typedef struct
      size_t
                   re_nsub;
      unsigned long re_flags;
                   *[4];
      void
} regex_t;
typedef ssize_t regoff_t;
typedef struct
      regoff_t
                   rm_so;
      regoff_t
                   rm_eo;
} regmatch_t;
```

Figure 6-56: <sys/resource.h>

```
#define RLIMIT_CPU
                          0
#define RLIMIT_FSIZE
                          1
#define RLIMIT_DATA
                           2
#define RLIMIT_STACK
                           3
#define RLIMIT_CORE
#define RLIMIT_NOFILE
                           5
#define RLIMIT_VMEM
                           6
#define RLIM_NLIMITS
                          RLIMIT_VMEM
#define RLIMIT_AS
                           0x7fffffff
#define RLIM_INFINITY
typedef unsigned long
                          rlim_t;
struct rlimit {
      rlim_t
                    rlim_cur;
      rlim_t
                    rlim_max;
};
```

6-48 LIBRARIES

Figure 6-57: c.h>, Part 1 of 16

```
#define bool_t
                          int
#define enum_t
                           int
enum xdr_op {
      XDR_ENCODE=0,
      XDR_DECODE=1,
      XDR_FREE=2
};
typedef bool_t
                    (*xdrproc_t)();
typedef struct {
      enum xdr_op x_op;
      struct xdr_ops {
             bool_t (*x_getlong)();
             bool_t (*x_putlong)();
             bool_t (*x_getbytes)();
             bool_t (*x_putbytes)();
             u_int (*x_getpostn)();
             bool_t (*x_setpostn)();
             long * (*x_inline)();
             void (*x_destroy)();
      } *x_ops;
      caddr_t
                   x_public;
      caddr_t
                   x_private;
      caddr_t
                   x_base;
      int
                    x_handy;
} XDR;
```

Figure 6-58: c.h>, Part 2 of 16

```
#define xdr_getpos(xdrs)
      (*(xdrs)->x_ops->x_getpostn)(xdrs)
#define xdr_setpos(xdrs, pos)
      (*(xdrs)->x_ops->x_setpostn)(xdrs, pos)
#define xdr_inline(xdrs, len)
      (*(xdrs)->x_ops->x_inline)(xdrs, len)
                                                                M
#define xdr_destroy(xdrs)
                                                                M
      (*(xdrs)->x_ops->x_destroy)(xdrs)
#define NULL_xdrproc_t
                         ((xdrproc_t)0)
struct xdr_discrim {
      int
                   value;
      xdrproc_t
                   proc;
};
```

6-50 LIBRARIES

Figure 6-59: c.h>, Part 3 of 16

```
#define MAX_AUTH_BYTES
                                 400
#define MAXNETNAMELEN
                                 255
#define HEXKEYBYTES
                                 48
enum auth_stat {
      AUTH_OK=0,
      AUTH_BADCRED=1,
      AUTH_REJECTEDCRED=2,
      AUTH_BADVERF=3,
      AUTH_REJECTEDVERF=4,
      AUTH_TOOWEAK=5,
      AUTH_INVALIDRESP=6,
      AUTH_FAILED=7
};
typedef u_long
                                                            M
                    u_int32
union des_block {
      struct {
             u_int32
                                 high;
                                                            M
                                                            M
             u_int32
                                 low;
       } key;
      char c[8];
};
typedef union des_block
                                 des_block;
```

Figure 6-60: c.h>, Part 4 of 16

```
struct opaque_auth {
      enum_t
                   oa_flavor;
      caddr_t
                   oa_base;
      u_int
                   oa_length;
};
typedef struct {
      struct opaque_auth ah_cred;
      struct opaque_auth ah_verf;
      union des_block
                          ah_key;
      struct auth_ops {
             void (*ah nextverf)();
             int
                   (*ah_marshal)();
             int
                  (*ah_validate)();
             int
                   (*ah_refresh)();
             void (*ah_destroy)();
      } *ah_ops;
      caddr_t
                   ah_private;
} AUTH;
#define auth_destroy(auth)
             ((*((auth)->ah_ops->ah_destroy))(auth))
#define AUTH_NONE
                          0
#define AUTH_NULL
                          0
#define AUTH_SYS
                          1
#define AUTH_UNIX
                          AUTH_SYS
#define AUTH_SHORT
                          2
#define AUTH_DES
#define AUTH_ESV
                          200004
```

M

6-52 LIBRARIES

Figure 6-61: c.h>, Part 5 of 16

```
enum clnt_stat {
      RPC_SUCCESS=0,
      RPC_CANTENCODEARGS=1,
      RPC_CANTDECODERES=2,
      RPC_CANTSEND=3,
      RPC_CANTRECV=4,
      RPC_TIMEDOUT=5,
      RPC_INTR=18,
      RPC_UDERROR=23,
      RPC_VERSMISMATCH=6,
      RPC_AUTHERROR=7,
      RPC PROGUNAVAIL=8,
      RPC_PROGVERSMISMATCH=9,
      RPC_PROCUNAVAIL=10,
      RPC_CANTDECODEARGS=11,
      RPC_SYSTEMERROR=12,
      RPC_UNKNOWNHOST=13,
      RPC_UNKNOWNPROTO=17,
      RPC_UNKNOWNADDR=19,
      RPC_NOBROADCAST=21,
      RPC_RPCBFAILURE=14,
      RPC_PROGNOTREGISTERED=15,
      RPC_N2AXLATEFAILURE=22,
      RPC_TLIERROR=20,
      RPC_FAILED=16
};
#define RPC_PMAPFAILURE
                          RPC_RPCBFAILURE
```

Figure 6-62: c.h>, Part 6 of 16

```
struct rpc_err {
      enum clnt_stat
                          re_status;
      union {
             struct {
                    int
                          errno;
                    int
                          t_errno;
             } RE_err;
             enum auth_stat
                                 RE_why;
             struct {
                    u_long
                                 low;
                                 high;
                    u_long
             } RE_vers;
             struct {
                    long s1;
                    long s2;
             } RE_lb;
      } ru;
};
typedef struct {
      AUTH
             *cl_auth;
      struct clnt_ops {
             enum clnt_stat
                               (*cl_call)();
             void
                          (*cl_abort)();
             void
                          (*cl_geterr)();
             bool_t
                          (*cl_freeres)();
             void
                          (*cl_destroy)();
             bool_t
                          (*cl_control)();
      } *cl_ops;
      caddr_t
                          cl_private;
      char
                          *cl_netid;
      char
                           *cl_tp;
} CLIENT;
```

6-54 LIBRARIES

Figure 6-63: c.h>, Part 7 of 16

```
#define FEEDBACK REXMIT1
#define FEEDBACK_OK
                          2.
#define clnt_call(rh, proc, xargs, argsp, xres, resp, secs)\
      ((*(rh)->cl_ops->cl_call)
      (rh, proc, xargs, argsp, xres, resp, secs))
#define clnt_abort(rh)
      ((*(rh)->cl_ops->cl_abort)(rh))
#define clnt_geterr(rh, errp)
      ((*(rh)->cl_ops->cl_geterr)(rh, errp))
#define clnt_freeres(rh,xres,resp)
      ((*(rh)->cl_ops->cl_freeres)(rh,xres,resp))
#define clnt_control(cl, rq, in)
      ((*(cl)->cl_ops->cl_control)(cl, rq, in))
#define clnt_destroy(rh)
      ((*(rh)->cl_ops->cl_destroy)(rh))
#define CLSET TIMEOUT
                                 1
#define CLGET_TIMEOUT
                                 2
#define CLGET_SERVER_ADDR
                                 3
#define CLGET_FD
                                 6
#define CLGET_SVC_ADDR
                                 7
#define CLSET FD CLOSE
#define CLSET_FD_NCLOSE
                                 9
#define CLSET_RETRY_TIMEOUT
#define CLGET_RETRY_TIMEOUT
                                 5
```

Figure 6-64: c.h>, Part 8 of 16

6-56 LIBRARIES

Figure 6-65: c.h>, Part 9 of 16

```
enum msg_type {
      CALL=0,
      REPLY=1
};
enum reply_stat {
      MSG_ACCEPTED=0,
      MSG_DENIED=1
};
enum accept_stat {
      SUCCESS=0,
      PROG_UNAVAIL=1,
      PROG_MISMATCH=2,
      PROC_UNAVAIL=3,
      GARBAGE_ARGS=4,
      SYSTEM_ERR=5
};
enum reject_stat {
      RPC_MISMATCH=0,
      AUTH_ERROR=1
};
```

Figure 6-66: c.h>, Part 10 of 16

```
struct accepted_reply {
      struct opaque_auth
                                  ar_verf;
      enum accept_stat
                                  ar_stat;
      union {
             struct {
                    u_long
                                  low;
                    u_long
                                  high;
             } AR_versions;
             struct {
                    caddr_t
                                  where;
                    xdrproc_t
                                  proc;
             } AR_results;
       } ru;
};
struct rejected_reply {
      enum reject_stat
                          rj_stat;
      union {
             struct {
                    u_long
                                  low;
                    u_long
                                  high;
             } RJ_versions;
             enum auth_stat
                                  RJ_why;
       } ru;
};
```

6-58 LIBRARIES

Figure 6-67: c.h>, Part 11 of 16

```
struct reply_body {
      enum reply_stat
                           rp_stat;
      union {
             struct accepted_reply RP_ar;
             struct rejected_reply RP_dr;
       } ru;
};
struct call_body {
      u\_long
                           cb_rpcvers;
      u_long
                           cb_prog;
      u_long
                           cb_vers;
      u\_long
                           cb_proc;
      struct opaque_auth cb_cred;
      struct opaque_auth cb_verf;
};
struct rpc_msg {
      u_long
                           rm_xid;
      enum msg_type
                           rm_direction;
      union {
             struct call_body
                                 RM_cmb;
             struct reply_body
                                 RM_rmb;
       } ru;
};
```

Figure 6-68: c.h>, Part 12 of 16

Figure 6-69: c.h>, Part 13 of 16

```
enum authdes_namekind {
      ADN FULLNAME,
      ADN_NICKNAME
};
struct authdes_fullname {
      char
                           *name;
      des_block
                           key;
                           window;
      u_long
};
struct authdes_cred {
      enum authdes namekind
                                 adc namekind;
      struct authdes_fullname
                                 adc_fullname;
      u_long
                                 adc_nickname;
};
```

6-60 LIBRARIES

Figure 6-70: c.h>, Part 14 of 16

```
enum xprt_stat {
      XPRT_DIED,
      XPRT_MOREREQS,
      XPRT_IDLE
};
typedef struct {
      int
                          xp_fd;
      u_short
                           xp_port;
      struct xp_ops {
             bool_t
                           (*xp_recv)();
             enum xprt_stat
                                (*xp_stat)();
             bool_t
                         (*xp_getargs)();
             bool_t
                           (*xp_reply)();
             bool_t
                           (*xp_freeargs)();
             void
                           (*xp_destroy)();
      } *xp_ops;
      int
                           xp_addrlen;
      char
                           *xp_tp;
      char
                           *xp_netid;
      struct netbuf
                          xp_ltaddr;
      struct netbuf
                          xp_rtaddr;
                           xp_raddr[16];
      struct opaque_auth xp_verf;
      caddr_t
                          xp_p1;
      caddr_t
                          xp_p2;
      caddr_t
                          xp_p3;
      int
                          xp_type;
} SVCXPRT;
```

M

System Data Interfaces

Figure 6-71: c.h>, Part 15 of 16

```
#define svc_getrpccaller(x) (&(x)->xp_rtaddr)
#define svc_getargs(xprt, xargs, argsp)
      (*(xprt)->xp_ops->xp_getargs)((xprt), (xargs), (argsp))
#define svc_freeargs(xprt, xargs, argsp)
      (*(xprt)->xp_ops->xp_freeargs)((xprt), (xargs), (argsp))
#define svc_destroy(xprt)
      (*(xprt)->xp_ops->xp_destroy)(xprt)
struct svc_req {
      u_long
                         rq_prog;
      u_long
                        rq_vers;
      u long
                        rq proc;
      struct opaque_auth rq_cred;
      caddr t
                         rq_clntcred;
      SVCXPRT
                         *rq_xprt;
};
                                                                   M
#define FD_SETSIZE
                       1024
#define NBBY 8
                                                                   M
                                                                   M
typedef long fd_mask;
#define NFDBITS (sizeof(fd mask) * NBBY)
                                                                   M
#define howmany(x, y) (((x)+((y)-1))/(y))
                                                                   M
                                                                   M
typedef struct fd_set {
       fd_mask fds_bits[howmany(FD_SETSIZE, NFDBITS)];
                                                                   M
                                                                   M
} fd_set;
extern fd_set svc_fdset;
```

6-62 LIBRARIES

Figure 6-72: c.h>, Part 16 of 16

```
struct rpcb {
      u_long
                    r_prog;
      u_long
                    r_vers;
      char
                    *r_netid;
      char
                    *r_addr;
      char
                    *r_owner;
};
typedef struct rpcb RPCB;
struct rpcblist {
      RPCB
                           rpcb_map;
                           *rpcb_next;
      struct rpcblist
};
```

Figure 6-73: <rtpriocntl.h>*

```
typedef struct rtparms {
            short rt_pri;
            ulong rt_tqsecs;
            long rt_tqnsecs;
} rtparms_t;

typedef struct rtinfo {
            short rt_maxpri;
} rtinfo_t;

#define RT_NOCHANGE -1
#define RT_TQINF -2
#define RT_TQDEF -3
```

Figure 6-74: <search.h>

```
typedef enum { FIND, ENTER } ACTION;
typedef struct entry { char *key; void *data; } ENTRY;
typedef enum { preorder, postorder, endorder, leaf } VISIT;
```

6-64 LIBRARIES

Figure 6-75: <sys/sem.h>

```
#define SEM UNDO
                    010000
#define GETNCNT
                    3
#define GETPID
                    4
#define GETVAL
                    5
#define GETALL
#define GETZCNT
                    7
#define SETVAL
                    8
#define SETALL
                    9
struct semid_ds {
      struct ipc_perm
                           sem_perm;
      struct sem
                    *sem base;
      ushort
                    sem_nsems;
      time_t
                    sem_otime;
      long
                    sem_pad1;
      time_t
                    sem_ctime;
                    sem_pad2;
      long
      long
                    sem_pad3[4];
};
struct sem {
      ushort
                    semval;
      pid_t
                    sempid;
      ushort
                    semncnt;
      ushort
                    semzcnt;
};
struct sembuf {
      ushort
                    sem_num;
      short
                    sem_op;
      short
                    sem_flg;
};
```

Figure 6-76: <setjmp.h>

Figure 6-77: <sys/shm.h>

```
#define SHMLBA
                           ((1) << 12)
#define SHM_RDONLY
                           010000
#define SHM_RND
                           020000
struct shmid_ds {
      struct ipc_perm
                           shm_perm;
      int
                    shm_segsz;
      struct anon_map
                           *shm_amp;
      ushort
                    shm_lkcnt;
      pid_t
                    shm_lpid;
                    shm_cpid;
      pid_t
      ulong
                    shm_nattch;
                    shm_cnattch;
      ulong
                    shm_atime;
      time_t
                    shm pad1;
      long
      time_t
                    shm_dtime;
      long
                    shm_pad2;
      time_t
                    shm_ctime;
      long
                    shm_pad3;
      long
                    shm_pad4[4];
};
```

6-66 LIBRARIES

Figure 6-78: <signal.h>, Part 1 of 3

#define	SIGHUP	1
#define	SIGINT	2
#define	SIGQUIT	3
#define	SIGILL	4
#define	SIGTRAP	5
#define	SIGIOT	6
#define	SIGABRT	6
#define	SIGEMT	7
#define	SIGFPE	8
#define	SIGKILL	9
#define	SIGBUS	10
#define	SIGSEGV	11
#define	SIGSYS	12
#define	SIGPIPE	13
#define	SIGALRM	14
#define	SIGTERM	15
#define	SIGUSR1	16
#define	SIGUSR2	17
#define	SIGCLD	18
#define	SIGCHLD	18
#define	SIGPWR	19
#define	SIGWINCH	20
#define	SIGURG	21
#define	SIGPOLL	22
#define	SIGIO	22
#define	SIGSTOP	23
#define	SIGTSTP	24
#define	SIGCONT	25
#define	SIGTTIN	26
#define	SIGTTOU	27
#define	SIGVTALRM	28
#define	SIGPROF	29

Figure 6-79: <signal.h>, Part 2 of 3

```
#define SIGXCPU
                                  30
#define SIGXFSZ
                                  31
#define SIG_DFL
                           (void(*)())0
#define SIG_ERR
                           (void(*)())-1
#define SIG_IGN
                           (void(*)())1
#define SIG_HOLD
                           (void(*)())2
#define SIG_BLOCK
#define SIG_UNBLOCK
                           2
#define SIG_SETMASK
typedef struct {
      unsigned int
                          sa_sigbits[4];
} sigset_t;
struct sigaction {
      int
                    sa_flags;
      void
                    (*sa_handler)();
      sigset_t
                    sa_mask;
                    sa_resv[2];
      int
};
#define SA_NOCLDSTOP
                           0x00020000
#define SA_ONSTACK
                           0x0000001
#define SA_RESETHAND
                           0 \times 00000002
#define SA_RESTART
                           0 \times 000000004
#define SA_SIGINFO
                           0x00000008
#define SA_NODEFER
                           0x0000010
#define SA_NOCLDWAIT
                           0x00010000
```

M

M

M

6-68 LIBRARIES

Figure 6-80: <signal.h>, Part 3 of 3

Figure 6-81: <sys/siginfo.h>, Part 1 of 5

```
#define ILL_ILLOPC
                           1
                           2
#define ILL_ILLOPN
#define ILL_ILLADR
                           3
#define ILL_ILLTRP
                           4
                           5
#define ILL_PRVOPC
#define ILL_PRVREG
                           6
                           7
#define ILL_COPROC
#define ILL_BADSTK
                           8
#define FPE_INTDIV
                           1
#define FPE INTOVF
                           2
                           3
#define FPE_FLTDIV
                           4
#define FPE_FLTOVF
                           5
#define FPE_FLTUND
#define FPE_FLTRES
                           6
#define FPE_FLTINV
                           7
#define FPE_FLTSUB
                           8
```

Figure 6-82: <sys/siginfo.h>, Part 2 of 5

#define SEGV_MAPERR	1
#define SEGV_ACCERR	2
#define BUS ADRALN	1
#define BUS ADRERR	2
_ ··	3
#define BUS_OBJERR	3
#define TRAP_BRKPT	1
#define TRAP TRACE	2
_	
#define CLD_EXITED	1
#define CLD_KILLED	2
#define CLD_DUMPED	3
#define CLD_TRAPPED	4
#define CLD_STOPPED	5
#define CLD_CONTINUED	6
	_
#define POLL_IN	1
#define POLL_OUT	2
#define POLL_MSG	3
#define POLL_ERR	4
#define POLL_PRI	5
#define POLL_HUP	6
#define SI MAXSZ	128
#define SI PAD	((SI MAXSZ / sizeof(int)) - 3)
1,4621116 61_11	((21_11101 / 511001(1110)) 5/

6-70 LIBRARIES

Figure 6-83: <sys/siginfo.h>, Part 3 of 5

```
typedef struct siginfo {
             si_signo;
      int
      int
             si_code;
      int
             si_errno;
      union {
                    _pad[SI_PAD];
             int
             struct {
                    pid_t _pid;
                    union {
                           struct {
                                  uid_t _uid;
                           } _kill;
                           struct {
                                                _utime;
                                  clock_t
                                               _status;
                                  int
                                  clock_t
                                               _stime;
                           } _cld;
                    } _pdata;
             } _proc;
             struct {
                    caddr_t
                                  _addr;
             } _fault;
             struct {
                    int
                           _fd;
                    long
                          _band;
             } _file;
      } _data;
} siginfo_t;
```

Figure 6-84: <sys/siginfo.h>, Part 4 of 5

```
#define si pid
                          _data._proc._pid
#define si_status
                          _data._proc._pdata._cld._status
#define si_stime
                          _data._proc._pdata._cld._stime
#define si_utime
                          _data._proc._pdata._cld._utime
#define si_uid
                          _data._proc._pdata._kill._uid
#define si_addr
                          _data._fault._addr
#define si_fd
                          _data._file._fd
                          _data._file._band
#define si_band
```

Figure 6-85: <sys/siginfo.h>*, Part 5 of 5

```
union sigval {
        int
                sival int;
        void
                *sival_ptr;
};
union notifyinfo {
        int
                        nisigno;
        void
                         (*nifunc)(union sigval);
};
struct sigevent {
        int
                         sigev_notify;
        union notifyinfo sigev notifyinfo;
        union sigval
                        sigev_value;
};
#define SIGEV_NONE
#define SIGEV_SIGNAL
                         2
#define SIGEV_CALLBACK
```

6-72 LIBRARIES

Figure 6-86: <sys/stat.h>, Part 1 of 2

```
#define _ST_FSTYPSZ
                          16
struct stat {
      dev_t
                    st_dev;
      long
                    st_pad1[3];
                    st_ino;
      ino_t
      mode_t
                    st_mode;
      nlink_t
                    st_nlink;
      uid_t
                    st_uid;
      gid_t
                    st_gid;
                    st_rdev;
      dev_t
                    st_pad2[2];
      long
      off_t
                    st_size;
      long
                    st_pad3;
      timestruc_t st_atim;
      timestruc_t st_mtim;
      timestruc_t st_ctim;
      long
                    st_blksize;
      long
                    st_blocks;
      char
                    st_fstype[_ST_FSTYPSZ];
      long
                    st_pad4[8];
};
#define st_atime
                    st_atim.tv_sec
#define st_mtime
                    st_mtim.tv_sec
#define st_ctime
                    st_ctim.tv_sec
```

Figure 6-87: <sys/stat.h>, Part 2 of 2

#define	S_IFMT	0xF000
#define	S_IFIFO	0x1000
#define	S_IFCHR	0x2000
#define	S_IFDIR	0x4000
#define	S_IFBLK	0x6000
#define	S_IFREG	0x8000
#define	S_IFLNK	0xA000
#define	S_ISUID	0x800
#define	S_ISGID	0x400
#define	S_ISVTX	0x200
#define		00700
#define	S_IRUSR	00400
#define	S_IWUSR	00200
#define	S_IXUSR	00100
#define		00070
#define	S_IRGRP	00040
#define	S_IWGRP	00020
#define	S_IXGRP	00010
#define	S_IRWXO	00007
#define	S_IROTH	00004
#define	S_IWOTH	00002
#define	S_IXOTH	00001
#define	S_ISFIFO(mode)	((mode&0xF000) == 0x1000)
		((mode&0xF000) == 0x2000)
#define	S_ISDIR(mode)	((mode&0xF000) == 0x4000)
#define	S_ISBLK(mode)	((mode&0xF000) == 0x6000)
#define	S_ISREG(mode)	((mode&0xF000) == 0x8000)

6-74 LIBRARIES

Figure 6-88: <sys/statvfs.h>

```
#define FSTYPSZ
                    16
typedef struct statvfs {
      u_long
                    f_bsize;
      u_long
                    f_frsize;
      u_long
                    f_blocks;
      u_long
                    f_bfree;
                    f_bavail;
      u_long
      u_long
                    f_files;
      u_long
                    f_ffree;
                    f_favail;
      u_long
                    f_fsid;
      u_long
      char
                    f_basetype[FSTYPSZ];
      u_long
                    f_flag;
      u_long
                    f_namemax;
      char
                    f_fstr[32];
      u_long
                    f_filler[16];
} statvfs_t;
#define ST_RDONLY
                    0x01
#define ST_NOSUID
                    0x02
#define ST_NOTRUNC 0x04
```

Figure 6-89: <stdarg.h>

```
G
typedef void
                              *va_list;
                                                                             G
extern void
                               va_end(va_list);
                                                                             G
#define va_start(list, name) (void) (list = (void *)((char *)&...))
                                                                             G
#define va_arg(list, mode)
                              ((mode *)(list = (char *)list + \setminus
                                                                             G
                               sizeof(mode)))[-1]
                                                                             G
#define va_end(list)
                              (void)0
```

NOTE

The construction &... is a syntactic extension to ANSI C and may not be supported by all C compilers. The intended semantics are to set list to the address on the stack of the first incoming argument in the variable part of the argument list. See "Function Calling Sequence" in Chapter 3.

G G

Figure 6-90: <stddef.h>

6-76 LIBRARIES

Figure 6-91: <stdio.h>, Part 1 of 2

typedef typedef	unsigned int long	size_t; fpos_t;
#define	NULL	0
#define	BUFSIZ	1024
#define	_IOFBF	0000
#define	_IOLBF	0100
#define	_IONBF	0004
#define	_IOEOF	0020
#define	_IOERR	0040
#define	EOF	(-1)
#define	FOPEN MAX	60
#define	FILENAME_MAX	1024
#define	L_ctermid	9
#define	L_cuserid	9
#define	P_tmpdir	"/var/tmp/"
#define	L_tmpnam	25
#define	stdin	(&iob[0])
#define	stdout	(&iob[1])
#define	stderr	(&iob[2])

Figure 6-92: <stdio.h>, Part 2 of 2

```
typedef struct {
      int
                          _cnt;
      unsigned char
                          * ptr;
      unsigned char
                          *_base;
      unsigned char
                          _flag;
      unsigned char
                          _file; tt
} FILE;
extern FILE
                    __iob[];
                                                                     G
#define clearerr(p) ((void)((p)->_flag &= ~(_IOERR | _IOEOF))) †
#define feof(p)
                   ((p)-> flaq \& IOEOF)
#define ferror(p) ((p)->_flag & _IOERR)
#define fileno(p) (p)->_file †
/* Non reentrant */
                                                                     M
                           (--(p)->_{cnt} < 0 ? _ _filbuf(p)
                                                                     M
#define getc_unlocked(p)
                                                                     M
             : (int)*(p)->_ptr++)
\#define\ putc\_unlocked(x, p) (--(p)->\_cnt < 0 ? \_ _flsbuf(x, p)
                                                                     M
             : (int)(*(p)->_ptr++ = (x)))
                                                                     M
                                                                     M
#define getchar_unlocked()
                           getc_unlocked(stdin)
#define putchar_unlocked(x) putc_unlocked((x), stdout)
                                                                     M
#define getc(p)
                  (--(p)->_cnt < 0 ? _ _filbuf(p)
                                                                     M
             : (int)*(p)->_ptr++)
                                                                     M
                                                                     M
\#define\ putc(x, p) \quad (--(p)->\_cnt < 0 ? \_ \_flsbuf(x, p)
             : (int)(*(p)->_ptr++ = (x)))
                                                                     M
#define getchar()
                                                                     M
                    getc(stdin)
#define putchar(x) putc((x), stdout)
                                                                     M
/* Reentrant versions available as functions only */
                                                                     M
```

† These macro definitions are moved to Level 2 as of January 1, 1993.

†† The _file member of the FILE struct is moved to Level 2 as of January 1, 1993.

6-78 LIBRARIES

NOTE

The macros clearerr, and fileno will be removed as a source interface in a future release supporting multi-processing. Applications should transition to the function equivalents of these macros in libc. Binary portability will be supported for existing applications.



The constant _NFILE has been removed. It should still appear in stdio.h, but may be removed in a future version of the header file. Applications may not be able to depend on fopen() failing on an attempt to open more than _NFILE files.

Figure 6-93: <stdlib.h>

```
typedef struct {
      int
             quot;
             rem;
} div_t;
typedef struct {
      long quot;
      long
           rem;
} ldiv_t;
typedef unsigned int
                          size_t;
#define NULL
                          0
#define EXIT_FAILURE
                          1
#define EXIT_SUCCESS
#define RAND_MAX
                          32767
extern unsigned char
                          __ctype[];
#define MB_CUR_MAX
                          (int)__ctype[520]
```

System Data Interfaces

6-79

Figure 6-94: <stropts.h>, Part 1 of 6

#define	SNDZERO	0x001
#define	SNDPIPE	0x002
#define	RNORM	0x000
#define	RMSGD	0x001
#define	RMSGN	0x002
#define	RMODEMASK	0x003
#define	RPROTDAT	0x004
#define	RPROTDIS	800x0
#define	RPROTNORM	0x010
#define	FLUSHR	0x01
#define	FLUSHW	0x02
#define	FLUSHRW	0x03
#define	FLUSHBAND	0x04

6-80 LIBRARIES

Figure 6-95: <stropts.h>, Part 2 of 6

#define	S_INPUT	0x0001
#define	S_HIPRI	0x0002
#define	S_OUTPUT	0x0004
#define	S_MSG	0x0008
#define	S_ERROR	0x0010
#define	S_HANGUP	0x0020
#define	S_RDNORM	0x0040
#define	S_WRNORM	S_OUTPUT
#define	S_RDBAND	0x0080
#define	S_WRBAND	0x0100
#define	S_BANDURG	0x0200
#define	RS_HIPRI	0x01
#define	MSG_HIPRI	0x01
#define	MSG_ANY	0x02
#define	MSG_BAND	0x04
#define	MORECTL	1
#define	MOREDATA	2
#define	MUXID_ALL	(-1)
	ANYMARK	0x01
#define	LASTMARK	0x02

Figure 6-96: <stropts.h>, Part 3 of 6

#define	STR	('S'<<8)
#define	I_NREAD	(STR 01)
#define	I_PUSH	(STR 02)
#define	I_POP	(STR 03)
#define	I_LOOK	(STR 04)
#define	I_FLUSH	(STR 05)
#define	I_SRDOPT	(STR 06)
#define	I_GRDOPT	(STR 07)
#define	I_STR	(STR 010)
#define	I_SETSIG	(STR 011)
#define	I_GETSIG	(STR 012)
#define	I_FIND	(STR 013)
#define	I_LINK	(STR 014)
#define	I_UNLINK	(STR 015)
#define	I_PEEK	(STR 017)
#define	I_FDINSERT	(STR 020)
#define	I_SENDFD	(STR 021)
#define	I_RECVFD	(STR 016)
#define	I_SWROPT	(STR 023)
#define	I_GWROPT	(STR 024)
#define	I_LIST	(STR 025)
#define	I_PLINK	(STR 026)
#define	I_PUNLINK	(STR 027)

6-82 LIBRARIES

Figure 6-97: <stropts.h>, Part 4 of 6

#define	I_FLUSHBAND	(STR 034)
#define	I_CKBAND	(STR 035)
#define	I_GETBAND	(STR 036)
#define	I_ATMARK	(STR 037)
#define	I_SETCLTIME	(STR 040)
#define	I_GETCLTIME	(STR 041)
#define	I_CANPUT	(STR 042)

Figure 6-98: <stropts.h>, Part 5 of 6

```
struct strioctl {
       int
              ic_cmd;
       int
              ic_timout;
       int
              ic_len;
       char
              *ic_dp;
};
struct strbuf {
       int
             maxlen;
       int
              len;
       char
             *buf;
};
struct strpeek {
       struct strbuf
                           ctlbuf;
       struct strbuf
                           databuf;
       long
                           flags;
};
struct strfdinsert {
       struct strbuf
                           ctlbuf;
       struct strbuf
                           databuf;
      long
                           flags;
       int
                           fildes;
       int
                           offset;
};
```

6-84 LIBRARIES

Figure 6-99: <stropts.h>, Part 6 of 6

```
struct strrecvfd {
      int
            fd;
      uid_t uid;
      gid_t gid;
      char fill[8];
};
#define FMNAMESZ
                    8
struct str_mlist {
      char l_name[FMNAMESZ+1];
};
struct str_list {
      int
                          sl_nmods;
                          *sl_modlist;
      struct str_mlist
};
struct bandinfo {
      unsigned char
                          bi_pri;
      int
                          bi_flag;
};
```

Figure 6-100: <synch.h>*, Part 1 of 3

```
#define USYNC THREAD
                       0
#define USYNC_PROCESS
                       1
typedef struct thrq_elt thrq_elt_t;
struct thrq_elt {
      thrq_elt_t *thrq_next;
      thrq_elt_t *thrq_prev;
};
typedef volatile struct {
      lwp_mutex_t m_lmutex;
      long
              m_type;
      lwp_mutex_t m_sync_lock;
      thrq_elt_t m_sleepq;
      long
                  filler;
} mutex_t;
typedef volatile struct {
      lwp_cond_t c_lcond;
      long
                   c_type;
      thrq_elt_t *c_syncq;
      lwp_mutex_t c_sync_lock;
} cond_t;
```

6-86 LIBRARIES

Figure 6-101: <synch.h>*, Part 2 of 3

```
typedef volatile struct {
      mutex_t
                    s_mutex;
      cond_t
                    s_cond;
      short
                    s_count;
      short
                    s_wakecnt;
      int
                    s_type;
} sema_t;
typedef volatile struct rwcv rwcv_t;
struct rwcv {
      cond_t rwcv_cond;
      rwcv_t *rwcv_next;
      char rwcv_rw;
      char rwcv_wakeup;
      short rwcv_readerwanted;
} ;
typedef volatile struct rwlock rwlock_t;
struct rwlock {
      mutex_t
                   rw_mutex;
      lwp_cond_t rw_lwpcond;
      int
                   rw_type;
      short
                   rw_readers;
      char
                   rw_writer;
      char
                   rw_wrwakeup;
      short
                   rw_writerwanted;
      short
                   rw_rdwakecnt;
      rwcv_t
                   *rw_cvqhead;
      rwcv_t
                   *rw_cvqtail;
      long
                   pad[4];
} ;
```

Figure 6-102: <synch.h>*, Part 3 of 3

```
typedef volatile struct {
      mutex_t
                   rm_mutex;
      pid_t
                   rm_pid;
      thread_t
                   rm_owner;
      int
                   rm_depth;
                   filler;
      long
} rmutex_t;
typedef volatile struct barrier_t;
struct barrier {
      mutex_t
                   b_lock;
      int
                   b_type;
      unsigned int b_count;
      unsigned int b_waiting;
      unsigned int b_generation;
      cond_t
                   b_cond;
} ;
```

Figure 6-103: <sys/sysi86.h>

#define	SI86FPHW	40
#define #define #define #define #define	FP_SW FP_HW FP_287	0 1 2 2 3

6-88 LIBRARIES

Figure 6-104: <termios.h>, Part 1 of 10

```
#define _POSIX_VDISABLE
#define CTRL(c)
                           ((c)&037)
#define IBSHIFT
                           16
#define NCC
#define NCCS
                           19
typedef unsigned long
                           tcflag_t;
typedef unsigned char
                           cc_t;
typedef unsigned long
                           speed_t;
struct termios {
      tcflag_t
                    c_iflag;
                    c_oflag;
      tcflag_t
      tcflag_t
                    c_cflag;
      tcflag_t
                    c_lflag;
      cc_t
                    c_cc[NCCS];
};
```

Figure 6-105: <termios.h>, Part 2 of 10

#define	VINTR	0
#define	VQUIT	1
#define	VERASE	2
#define	VKILL	3
#define	VEOF	4
#define	VEOL	5
#define	VEOL2	6
#define	VMIN	4
#define	VTIME	5
#define	VSWTCH	7
#define	VSTART	8
#define	VSTOP	9
#define	VSUSP	10
#define	VDSUSP	11
#define	VREPRINT	12
#define	VDISCARD	13
#define	VWERASE	14
#define	VLNEXT	15

6-90 LIBRARIES

Figure 6-106: <termios.h>, Part 3 of 10

#define	CNUL	0
#define	CDEL	0177
#define	CESC	'\\'
#define	CINTR	0177
#define	CQUIT	034
#define	CERASE	'# <i>'</i>
#define	CKILL	'@'
#define	CEOT	04
#define	CEOL	0
#define	CEOL2	0
#define	CEOF	04
#define	CSTART	021
#define	CSTOP	023
#define	CSWTCH	032
#define	CNSWTCH	0
#define	CSUSP	CTRL('z')
#define	CDSUSP	CTRL('y')
#define	CRPRNT	CTRL('r')
#define	CFLUSH	CTRL('o')
#define	CWERASE	CTRL('w')
#define	CLNEXT	CTRL('v')

Figure 6-107: <termios.h>, Part 4 of 10

#define	IGNBRK	0000001
#define	BRKINT	0000002
#define	IGNPAR	0000004
#define	PARMRK	0000010
#define	INPCK	0000020
#define	ISTRIP	0000040
#define	INLCR	0000100
#define	IGNCR	0000200
#define	ICRNL	0000400
#define	IUCLC	0001000
#define	IXON	0002000
#define	IXANY	0004000
#define	IXOFF	0010000
#define	IMAXBEL	0020000
#define	DOSMODE	0100000

6-92 LIBRARIES

Figure 6-108: <termios.h>, Part 5 of 10

#define	OPOST	0000001
#define	OLCUC	0000002
#define	ONLCR	0000004
#define	OCRNL	0000010
#define	ONOCR	0000020
#define	ONLRET	0000040
#define	OFILL	0000100
#define	OFDEL	0000200
#define	NLDLY	0000400
#define	NL0	0
#define	NL1	0000400
#define	CRDLY	0003000
#define	CR0	0
#define	CR1	0001000
#define	CR2	0002000
#define	CR3	0003000
#define	TABDLY	0014000

Figure 6-109: <termios.h>, Part 6 of 10

_			
	#define	TAB0	0
	#define	TAB1	0004000
	#define	TAB2	0010000
	#define	TAB3	0014000
	#define	XTABS	0014000
	#define	BSDLY	0020000
	#define	BS0	0
	#define	BS1	0020000
	#define	VTDLY	0040000
	#define	VT0	0
	#define	VT1	0040000
	#define	FFDLY	0100000
	#define	FF0	0
	#define	FF1	0100000
	#define	PAGEOUT	0200000
	#define	WRAP	0400000

6-94 LIBRARIES

Figure 6-110: <termios.h>, Part 7 of 10

#define	CBAUD	0000017
#define	CSIZE	0000060
#define	CS5	0
#define	CS6	0000020
#define	CS7	0000040
#define	CS8	0000060
#define	CSTOPB	0000100
#define	CREAD	0000200
#define	PARENB	0000400
#define	PARODD	0001000
#define	HUPCL	0002000
#define	CLOCAL	0004000
#define	RCV1EN	0010000
#define	XMT1EN	0020000
#define	LOBLK	0040000
#define	XCLUDE	0100000
#define	CIBAUD	03600000
#define	PAREXT	04000000

Figure 6-111: <termios.h>, Part 8 of 10

#define	ISIG	0000001
#define	ICANON	0000002
#define	XCASE	0000004
#define	ECHO	0000010
#define	ECHOE	0000020
#define	ECHOK	0000040
#define	ECHONL	0000100
#define	NOFLSH	0000200
#define	TOSTOP	0000400
#define	ECHOCTL	0001000
#define	ECHOPRT	0002000
#define	ECHOKE	0004000
#define	DEFECHO	0010000
#define	FLUSHO	0020000
#define	PENDIN	0040000
#define	IEXTEN	0100000

6-96 LIBRARIES

Figure 6-112: <termios.h>, Part 9 of 10

#define	TIOC	('T'<<8)
#define	TCGETA	(TIOC 1)
#define	TCSETA	(TIOC 2)
#define	TCSETAW	(TIOC 3)
#define	TCSETAF	(TIOC 4)
#define	TCSBRK	(TIOC 5)
#define	TCXONC	(TIOC 6)
#define	TCFLSH	(TIOC 7)
	TIOCGWINSZ TIOCSWINSZ	(TIOC 104) (TIOC 103)
#define #define #define		(TIOC 13) (TIOC 14) (('T'<<8) 14) (TIOC 15) (('T'<<8) 15) (TIOC 16) (('T'<<8) 16)
#deline	ICONTINOU	((1 <<0) 10)

Figure 6-113: <termios.h>, Part 10 of 10

```
#define TCIFLUSH
                           0
#define TCOFLUSH
                           1
#define TCIOFLUSH
                           2
#define TCOOFF
                           0
#define TCOON
                           1
#define TCIOFF
                           2
#define TCION
                           3
#define B0
                           0
#define B50
                           1
#define B75
                           2
#define B110
                           3
#define B134
                           4
#define B150
                           5
#define B200
                           6
#define B300
                           7
#define B600
                           8
                           9
#define B1200
#define B1800
                           10
#define B2400
                           11
#define B4800
                           12
#define B9600
                           13
#define B19200
                           14
#define B38400
                           15
struct winsize {
      unsigned short
                           ws_row;
      unsigned short
                           ws_col;
      unsigned short
                           ws_xpixel;
      unsigned short
                           ws_ypixel;
};
```

6-98 LIBRARIES

Figure 6-114: <thread.h>*, Part 1 of 2

```
#define THR_SUSPENDED 0x1
#define THR_BOUND
                      0x2
#define THR_INCR_CONC 0x4
#define THR_DETACHED 0x8
#define THR_DAEMON
                      0x10
#define SCHED_TS
                      1
#define SCHED_OTHER
                      1
#define SCHED_FIFO
                      2
#define SCHED_RR
                      3
typedef
             id_t thread_t;
                                PC_CLPARMSZ
#define POLICY_PARAM_SZ
```

Figure 6-115: <thread.h>*, Part 2 of 2

```
typedef struct {
    id_t policy;
    long policy_params[POLICY_PARAM_SZ];
} sched_param_t;

struct ts_param {
    int prio;
};

struct fifo_param {
    int prio;
};

struct rr_param {
    int prio;
};

typedef unsigned int thread_key_t;
```

Figure 6-116: <sys/ticlts.h>

```
#define TCL_BADADDR 1
#define TCL_BADOPT 2
#define TCL_NOPEER 3
#define TCL_PEERBADSTATE 4
#define TCL_DEFAULTADDRSZ 4
```

6-100 LIBRARIES

Figure 6-117: <sys/ticots.h>

l	#define	TCO_NOPEER	ECONNREFUSED
l	#define	TCO_PEERNOROOMONQ	ECONNREFUSED
l	#define	TCO_PEERBADSTATE	ECONNREFUSED
l	#define	TCO_PEERINITIATED	ECONNRESET
l	#define	TCO_PROVIDERINITIATED	ECONNABORTED
l	#define	TCO_DEFAULTADDRSZ	4
l			

Figure 6-118: <sys/ticotsord.h>

#dofino	TCOO NOPEER	1
	-	_
	TCOO_PEERNOROOMONQ	2
#define	TCOO_PEERBADSTATE	3
#define	TCOO_PEERINITIATED	4
#define	TCOO_PROVIDERINITIATED	5
#define	TCOO_DEFAULTADDRSZ	4

NOTE

The sys/tihdr.h and sys/timod.h headers previously included in this document were unneccessary as they did not contain user level information and have therefore been removed from this document.

System Data Interfaces

```
Figure 6-119: <time.h>*
```

```
struct timespec {
    time_t tv_sec;
    long tv_nsec;
};
```

6-102 LIBRARIES

Figure 6-120: <sys/time.h>

```
typedef long clock_t;
typedef long time_t;
typedef unsigned int size_t;
typedef struct timespec {
      time_t
                 tv_sec;
      long
                   tv_nsec;
} timestruc_t;
#define CLOCKS_PER_SEC
                                 1000000
struct tm {
      int
             tm_sec;
      int
             tm_min;
      int
             tm_hour;
      int
             tm_mday;
      int
             tm_mon;
      int
             tm_year;
      int
             tm_wday;
      int
             tm_yday;
      int
             tm_isdst;
};
extern char *tzname[2];
#define CLK_TCK
                   _sysconf(3)
extern long timezone;
extern int daylight;
```

6-103

M

 $_{\mathrm{M}}^{\mathrm{M}}$

M M

```
Figure 6-121: <sys/times.h>
```

NOTE

This edition introduces the xti.h header which contains the same information as the current tiuser.h. The new xti.h header is a superset of the previous edition's tiuser.h.

Μ

Μ

Μ

tiuser.h has been moved to Level 2 and will be removed in future editions of the ABI. In the future xti.h should be used as a replacement for tiuser.h.

6-104 LIBRARIES

Figure 6-122: <tiuser.h>, Error Return Values

_			
	#define	TBADADDR	1
	#define	TBADOPT	2
	#define	TACCES	3
	#define	TBADF	4
	#define	TNOADDR	5
	#define	TOUTSTATE	6
	#define	TBADSEQ	7
	#define	TSYSERR	8
	#define	TLOOK	9
	#define	TBADDATA	10
	#define	TBUFOVFLW	11
	#define	TFLOW	12
	#define	TNODATA	13
		TNODIS	14
	#define	TNOUDERR	15
	#define	TBADFLAG	16
	#define	TNOREL	17
		TNOTSUPPORT	
	#define	TSTATECHNG	19
	#define	TNOSTRUCTYPE	20
	#define	TBADNAME	21
	#define	TBADQLEN	22
	#define	TADDRBUSY	23
			24
	#define	TPROVMISMATCH	25
	#define	TRESQLEN	26
	#define	TRESADDR	27
	#define	TQFULL	28
	#define	TPROTO	29

Figure 6-123: <tiuser.h>, Event Bitmasks

Figure 6-124: <tiuser.h>, Flags

#define	T_MORE	0x001
#define	T_EXPEDITED	0x002
#define	T_NEGOTIATE	0x004
#define	T_CHECK	0x008
#define	T_DEFAULT	0x010
#define	T_SUCCESS	0x020
#define	T_FAILURE	0x040

6-106 LIBRARIES

Figure 6-125: <tiuser.h>, Service Types

#define	T_COTS	01
#define	T_COTS_ORD	02
#define	T_CLTS	03

Figure 6-126: <tiuser.h>, Values for flags field in t_i nfo structure

#define	T_SENDZERO	0x000001	M

System Data Interfaces

Figure 6-127: <tiuser.h>, Transport Interface Data Structures, 1 of 2

```
struct t_info {
       long
             addr;
       long
             options;
             tsdu;
       long
       long
             etsdu;
       long
             connect;
       long
             discon;
       long
             servtype;
       long
             flags;
};
struct netbuf {
      unsigned int
                           maxlen;
      unsigned int
                           len;
       char
                            *buf;
};
struct t_bind {
       struct netbuf
                           addr;
       unsigned
                           qlen;
};
struct t_optmgmt {
       struct netbuf
                           opt;
       long
                           flags;
};
```

M

NOTE

Applications invoking TLI binary interfaces to t_open or t_getinfo will see the t_info structure without the flags member. Those applications invoking the XTI M versions of t_open or t_getinfo will see the t_info structure with the flags

6-108 LIBRARIES

Figure 6-128: <tiuser.h>, Transport Interface Data Structures, 2 of 2

```
struct t_discon {
      struct netbuf
                           udata;
       int
                           reason;
      int
                           sequence;
};
struct t_call {
      struct netbuf
                           addr;
      struct netbuf
                           opt;
      struct netbuf
                           udata;
      int
                           sequence;
};
struct t_unitdata {
      struct netbuf
                           addr;
      struct netbuf
                           opt;
      struct netbuf
                           udata;
};
struct t_uderr {
      struct netbuf
                           addr;
      struct netbuf
                           opt;
      long
                           error;
};
```

Figure 6-129: <tiuser.h>, Structure Types

#define	T_BIND	1
#define	T_OPTMGMT	2
#define	T_CALL	3
#define	T_DIS	4
#define	T_UNITDATA	5
#define	T_UDERROR	б
#define	T_INFO	7

Figure 6-130: <tiuser.h>, Fields of Structures

#define T_ADDR	0x01
#define T_OPT	0x02
#define T_UDATA	0x04
#define T_ALL	0xffff

M

NOTE

Differences between XTI and TLI have forced the value of T_ALL to change. M The previous edition's T_ALL value will not produce the same results as the new T_ALL .

6-110 LIBRARIES

Figure 6-131: <tiuser.h>, Transport Interface States

#define	T_UNINIT	0
#define	T_UNBND	1
#define	T_IDLE	2
#define	T_OUTCON	3
#define	T_INCON	4
#define	T_DATAXFER	5
#define	T_OUTREL	6
#define	T_INREL	7
#define	T_FAKE	8
#define	T_NOSTATES	9

Figure 6-132: <tiuser.h>, User-level Events

#define	T_OPEN	0
#define	T_BIND	1
#define	T_OPTMGMT	2
#define	T_UNBIND	3
#define	T_CLOSE	4
#define	T_SNDUDATA	5
#define	T_RCVUDATA	6
#define	T_RCVUDERR	7
#define	T_CONNECT1	8
#define	T_CONNECT2	9
#define	T_RCVCONNECT	10
#define	T_LISTN	11
#define	T_ACCEPT1	12
#define	T_ACCEPT2	13
#define	T_ACCEPT3	14
#define	T_SND	15
#define	T_RCV	16
#define	T_SNDDIS1	17
#define	T_SNDDIS2	18
#define	T_RCVDIS1	19
#define	T_RCVDIS2	20
#define	T_RCVDIS3	21
#define	T_SNDREL	22
#define	T_RCVREL	23
#define	T_PASSCON	24
#define	T_NOEVENTS	25

6-112 LIBRARIES

Figure 6-133: <tspriocntl.h>*

```
typedef struct tsparms {
        short ts_uprilim;
        short ts_upri;
} tsparms_t;

typedef struct tsinfo {
        short ts_maxupri;
} tsinfo_t;

#define TS_NOCHANGE -32768
```

Figure 6-134: <sys/types.h>

typedef	unsigned	char	uchar_t;
typedef	unsigned	short	ushort_t;
typedef	unsigned	int	uint_t;
typedef	unsigned	long	ulong_t;
typedef			caddr_t;
typedef	long		daddr_t;
typedef	long		off_t;
typedef			id_t;
typedef	int		key_t;
typedef	ulong_t		mode_t;
typedef	long		uid_t;
typedef			gid_t;
	ulong_t		nlink_t;
	ulong_t		dev_t;
	ulong_t		ino_t;
typedef	_		pid_t;
typedef			size_t;
typedef	_		time_t;
typedef	long		clock_t;
		_	_
	unsigned		ushort;
typedef	unsigned	long	ulong;
	2	-1	la •
	unsigned		u_char;
	unsigned		u_short;
	unsigned		u_int;
rypeaei	unsigned	TOHG	u_long;

6-114 LIBRARIES

Figure 6-135: <ucontext.h>, Part 1 of 2

typedef	int	greg_t;
#define	NGREG	19
typedef	greg_t	gregset_t[NGREG];
#define	SS	18
#define	UESP	17
#define	EFL	16
#define	CS	15
#define	EIP	14
#define	ERR	13
#define	TRAPNO	12
#define	EAX	11
#define	ECX	10
#define	EDX	9
#define	EBX	8
#define	ESP	7
#define	EBP	6
#define	ESI	5
#define	EDI	4
#define	DS	3
#define	ES	2
#define	FS	1
#define	GS	0

Figure 6-136: <ucontext.h>, Part 2 of 2

```
typedef struct fpregset {
      union {
             struct fpchip_state {
                   int
                        state[27];
                   int
                          status;
             } fpchip_state;
             struct fp_emul_space {
                   char fp_emul[246];
                   char fp_epad[2];
             } fp_emul_space;
                  f_fpregs[62];
             int
      } fp_reg_set;
      long f_wregs[33];
} fpregset_t;
typedef struct {
      gregset_t
                   gregs;
      fpregset_t
                   fpregs;
} mcontext_t;
typedef struct ucontext {
                   uc_flags;
      u_long
                          *uc link;
      struct ucontext
      sigset_t
                   uc_sigmask;
      stack_t
                   uc_stack;
      mcontext_t uc_mcontext;
      long
                   uc_filler[5];
} ucontext_t;
```

6-116 LIBRARIES

Figure 6-137: <sys/uio.h>

```
typedef struct iovec {
      caddr_t iov_base;
      int iov_len;
} iovec_t;
```

Figure 6-138: <ulimit.h>

#define	UL_GETFSIZE	1
#define	UL_SETFSIZE	2

Figure 6-139: <unistd.h>, Part 1 of 2

```
#define R_OK
                                 4
                                 2
#define W_OK
#define X_OK
                                 1
#define F_OK
                                 0
#define F_ULOCK
                                 0
#define F_LOCK
                                 1
#define F_TLOCK
                                  2
#define F_TEST
                                 3
#define SEEK_SET
                                 0
#define SEEK_CUR
                                 1
                                 2
#define SEEK_END
#define _SC_ARG_MAX
                                 1
#define _SC_CHILD_MAX
                                 2
#define _SC_CLK_TCK
                                  3
```

System Data Interfaces

6-117

Figure 6-139: <unistd.h>, Part 1 of 2 (continued)

1	#define	_SC_NGROUPS_MAX	4
	#define	_SC_OPEN_MAX	5
	#define	_SC_JOB_CONTROL	6
	#define	_SC_SAVED_IDS	7
	#define	_SC_VERSION	8
	#define	_SC_PASS_MAX	9
	#define	_SC_LOGNAME_MAX	10
	#define	_SC_PAGESIZE	11
	#define	_SC_XOPEN_VERSION	12
	#define	_CS_PATH	1
	#define	_CS_HOSTNAME	2
	#define	_CS_RELEASE	3
	#define	_CS_VERSION	4
	#define	_CS_MACHINE	5
	#define	_CS_ARCHITECTURE	6
	#define	_CS_HW_SERIAL	7
	#define	_CS_HW_PROVIDER	8
	#define	_CS_SRPC_DOMAIN	9
	#define	_CS_SYSNAME	11

6-118 LIBRARIES

Figure 6-140: <unistd.h>, Part 2 of 2

```
#define PC LINK MAX
                                 1
#define _PC_MAX_CANON
                                 2
#define _PC_MAX_INPUT
                                 3
                                 4
#define _PC_NAME_MAX
#define _PC_PATH_MAX
                                 5
#define _PC_PIPE_BUF
                                 6
#define _PC_NO_TRUNC
                                 7
                                 8
#define _PC_VDISABLE
#define _PC_CHOWN_RESTRICTED
#define _POSIX_JOB_CONTROL
                                 1
#define POSIX SAVED IDS
#define _POSIX_VDISABLE
                                 0
#define _POSIX_VERSION
#define _XOPEN_VERSION
/* starred values vary and should be
      retrieved using sysconf() or pathconf() */
                                 0
#define STDIN_FILENO
#define STDOUT_FILENO
                                 1
#define STDERR_FILENO
```

Figure 6-141: <utime.h>

```
struct utimbuf {
    time_t actime;
    time_t modtime;
};
```

Figure 6-142: <sys/utsname.h>

```
#define SYS_NMLN 257

struct utsname {
    char sysname[SYS_NMLN];
    char nodename[SYS_NMLN];
    char release[SYS_NMLN];
    char version[SYS_NMLN];
    char machine[SYS_NMLN];
};
```

6-120 LIBRARIES

Figure 6-143: <wait.h>

#define	WEXITED	0001
#define	WTRAPPED	0002
#define	WSTOPPED	0004
#define	WCONTINUED	0010
#define	WUNTRACED	0004
#define	WNOHANG	0100
#define	WNOWAIT	0200
#define	WCONTFLG	0177777
#define	WCOREFLG	0200
		(//
#define	WWORD(stat)	((int)((stat))&0177777)
#define	WSTOPFLG	0177
#define	WSIGMASK	0177
#define	WLOBYTE(stat) ((int)((stat)&0377))
#define	WHIBYTE(stat) ((int)(((stat)>>8)&0377))
#define	<pre>WIFEXITED(stat)</pre>	(WLOBYTE(stat)==0)
#define	WIFSIGNALED(stat)	(WLOBYTE(stat)>0&&WHIBYTE(stat)==0)
#define	<pre>WIFSTOPPED(stat)</pre>	(WLOBYTE(stat) == WSTOPFLG&&WHIBYTE(stat)!=0)
#define	WIFCONTINUED(stat)	(WWORD(stat) == WCONTFLG)
#define	WEXITSTATUS(stat)	WHIBYTE(stat)
	,	(WLOBYTE(stat)&WSIGMASK)
	WSTOPSIG(stat)	WHIBYTE(stat)
#actifie	MDIOEDIG (BCCC)	mildin beac,
#define	WCOREDUMP(stat)	((stat)&WCOREFLG)

```
Figure 6-144: <wchar.h>
```

6-122 LIBRARIES

Figure 6-145: <wctype.h>*, Part 1 of 3

typedef	_	wint_t;
typedef	unsigned	l long wctype_t;
#define	WEOF (-1	.)
#define	_U	01
#define	_L	02
#define	_N	04
#define	_S	010
#define	_P	020
#define	_C	040
#define	_B	0100
#define	_X	0200
#define	E1	0x00000100
#define	_E1 E2	
#define	_E2	
#define	E3	
#define	E5	
#define	_E5	
#define	_=°	
#define	 E8	
#define	E9	0x00010000
#define	_ _E1	.0 0x00020000
#define	_ _E1	.1 0x00040000
#define	_E1	.2 0x00080000
#define	_E1	.3 0x00100000
#define	_E1	.4 0x00200000
#define	_E1	.5 0x00400000
#define	_E1	.6 0x00800000
#define	_E1	.7 0x01000000
#define	_E1	.8 0x02000000
#define	_E1	.9 0x04000000
#define	_E2	
#define	_E2	21 0x10000000

Figure 6-146: <wctype.h>*, Part 2 of 3

```
(_U | _L | _N)
#define PD ALNUM
#define _PD_ALPHA
                    (_U | _L)
#define _PD_BLANK
                    (_B)
#define _PD_CNTRL
                    (_C)
#define _PD_DIGIT
                    (_N)
#define _PD_GRAPH
                    (_P | _U | _L | _N | _E1 |
                           _E2 | _E5 | _E6)
#define _PD_LOWER
                    (_L)
#define _PD_PRINT
                    (_P | _U | _L | _N | _B |
                           _E1 | _E2 | _E5 | _E6)
#define _PD_PUNCT
                    (_P)
#define PD SPACE
                    (S)
#define _PD_UPPER
                    (_U)
#define _PD_XDIGIT
                    (_X)
#define iswalnum(c)
                            __isw(c, _PD_ALNUM)
                           __isw(c, _PD_ALPHA)
#define iswalpha(c)
#define iswcntrl(c)
                           __isw(c, _PD_CNTRL)
#define iswdigit(c)
                           __isw(c, _PD_DIGIT)
#define iswgraph(c)
                           __isw(c, _PD_GRAPH)
                           __isw(c, _PD_LOWER)
#define iswlower(c)
#define iswprint(c)
                           __isw(c, _PD_PRINT)
#define iswpunct(c)
                           isw(c, PD PUNCT)
#define iswspace(c)
                           __isw(c, _PD_SPACE)
#define iswupper(c)
                           __isw(c, _PD_UPPER)
#define iswxdigit(c)
                           __isw(c, _PD_XDIGIT)
#define towlower(c)
                           __tow(c, _PD_UPPER)
                           __tow(c, _PD_LOWER)
#define towupper(c)
#define isphonogram(c)
                           __isx(c, _E1)
#define isideogram(c)
                           __isx(c, _E2)
#define isenglish(c)
                           __isx(c, _E3)
#define isnumber(c)
                           __isx(c, _E4)
#define isspecial(c)
                           __isx(c, _E5)
```

6-124 LIBRARIES

Figure 6-147: <wctype.h>*, Part 3 of 3

```
#define iscodeset0(c)
                           (((c) \& `(wchar t)0xff) == 0)
#define iscodeset1(c)
                           (((c) >> 28) == 0x3)
#define iscodeset2(c)
                           (((c) >> 28) == 0x1)
#define iscodeset3(c)
                           (((c) >> 28) == 0x2)
inline int __isw(wint_t c, wctype_t t){
      if (c > 255)
             return (__iswctype(c, t));
      return (1 + __ctype)[c] & t;
}
inline int __isx(wint_t c, wctype_t t){
      return (c > 255 && __iswctype(c, t));
inline wint_t __tow(wint_t c, wctype_t t){
       if (c > 255)
             return (__trwctype(c, t));
      if ((1 + \underline{ctype})[c] \& t)
             return (258 + __ctype)[c];
      return (c);
}
```

NOTE

The construction inline is a syntactic extension to ANSI C and may not be supported by all C compilers. The intended semantics are to behave like regular preprocessor function like macros except parameter names are local and expressions giving their initial values are evaluated exactly once.

System Data Interfaces

6-125

М

Μ

Figure 6-148: <wordexp.h>*

		1
#define WRDE_APPEND	0001	M
#define WRDE_DOOFFS	0002	M
#define WRDE_NOCMD	0004	M
#define WRDE_REUSE	0010	M
#define WRDE_SHOWERR	0020	M
#define WRDE_UNDEF	0040	M
#define WRDE NOSYS	(-1)	M
#define WRDE BADCHAR	(-2)	M
#define WRDE_BADVAL	(-3)	M
#define WRDE_CMDSUB	(-4)	M
#define WRDE_NOSPACE	(-5)	M
#define WRDE_SYNTAX	(-6)	M
typedef struct		M
{		M
size twe wordc;		M
char **we_wordv;		M
size_t we_offs;		M
} wordexp_t;		M
_		

6-126 LIBRARIES

X Window Data Definitions



This section is new to the Third Edition of this document, but will not be marked with the "G" diff-mark.

This section contains standard data definitions that describe system data for the optional X Window System libraries listed in the Generic ABI. These data definitions are referred to by their names in angle brackets: <name.h> and <sys/name.h>. Included in these data definitions are macro definitions and structure definitions. While an ABI-conforming system may provide X11 and X Toolkit Intrinsics interfaces, it need not contain the actual data definitions referenced here. Programmers should observe that the sources of the structures defined in these data definitions are defined in SVID or the appropriate X Consortium documentation (see chapter 10 in the Generic ABI).

System Data Interfaces

Figure 6-149: <X11/Atom.h>, Part 1 of 3

#define	XA_PRIMARY	((Atom)	1)
#define	XA_SECONDARY	((Atom)	2)
#define	XA_ARC	((Atom)	3)
#define	XA_ATOM	((Atom)	4)
#define	XA_BITMAP	((Atom)	5)
#define	XA_CARDINAL	((Atom)	6)
#define	XA_COLORMAP	((Atom)	7)
#define	XA_CURSOR	((Atom)	8)
#define	XA_CUT_BUFFER0	((Atom)	9)
#define	XA_CUT_BUFFER1	((Atom)	10)
#define	XA_CUT_BUFFER2	((Atom)	11)
#define	XA_CUT_BUFFER3	((Atom)	12)
#define	XA_CUT_BUFFER4	((Atom)	13)
#define	XA_CUT_BUFFER5	((Atom)	14)
#define	XA_CUT_BUFFER6	((Atom)	15)
#define	XA_CUT_BUFFER7	((Atom)	16)
#define	XA_DRAWABLE	((Atom)	17)
#define	XA_FONT	((Atom)	18)
#define	XA_INTEGER	((Atom)	19)
	XA_PIXMAP	((Atom)	20)
#define	XA_POINT	((Atom)	21)
#define	XA_RECTANGLE	((Atom)	
#define	XA_RESOURCE_MANAGER	((Atom)	23)
#define	XA_RGB_COLOR_MAP	((Atom)	24)
#define	XA_RGB_BEST_MAP	((Atom)	25)
#define	XA_RGB_BLUE_MAP	((Atom)	26)
#define	XA_RGB_DEFAULT_MAP	((Atom)	27)
#define	XA_RGB_GRAY_MAP	((Atom)	28)
#define	XA_RGB_GREEN_MAP	((Atom)	29)
	XA_RGB_RED_MAP	((Atom)	•
#define	XA_STRING	((Atom)	31)
#define	XA_VISUALID	((Atom)	32)

6-128 LIBRARIES

Figure 6-150: <X11/Atom.h>, Part 2 of 3

#define	XA_WINDOW	((Atom)	33)
#define	XA_WM_COMMAND	((Atom)	34)
#define	XA_WM_HINTS	((Atom)	35)
#define	XA_WM_CLIENT_MACHINE	((Atom)	36)
#define	XA_WM_ICON_NAME	((Atom)	37)
#define	XA_WM_ICON_SIZE	((Atom)	38)
#define	XA_WM_NAME	((Atom)	39)
#define	XA_WM_NORMAL_HINTS	((Atom)	40)
#define	XA_WM_SIZE_HINTS	((Atom)	41)
#define	XA_WM_ZOOM_HINTS	((Atom)	42)
#define	XA_MIN_SPACE	((Atom)	43)
#define	XA_NORM_SPACE	((Atom)	44)
#define	XA_MAX_SPACE	((Atom)	45)
#define	XA_END_SPACE	((Atom)	46)
#define	XA_SUPERSCRIPT_X	((Atom)	47)
#define	XA_SUPERSCRIPT_Y	((Atom)	48)
#define	XA_SUBSCRIPT_X	((Atom)	49)
#define	XA_SUBSCRIPT_Y	((Atom)	50)
#define	XA_UNDERLINE_POSITION	((Atom)	51)
#define	XA_UNDERLINE_THICKNESS	((Atom)	52)
#define	XA_STRIKEOUT_ASCENT	((Atom)	53)
#define	XA_STRIKEOUT_DESCENT	((Atom)	54)
#define	XA_ITALIC_ANGLE	((Atom)	55)
#define	XA_X_HEIGHT	((Atom)	56)
#define	XA_QUAD_WIDTH	((Atom)	57)
#define	XA_WEIGHT	((Atom)	58)
#define	XA_POINT_SIZE	((Atom)	59)
#define	XA_RESOLUTION	((Atom)	60)
#define	XA_COPYRIGHT	((Atom)	61)
#define	XA_NOTICE	((Atom)	62)
#define	XA_FONT_NAME	((Atom)	63)
#define	XA_FAMILY_NAME	((Atom)	64)

Figure 6-151: <X11/Atom.h>, Part 3 of 3

#define	XA_FULL_NAME	((Atom)	65)
#define	XA_CAP_HEIGHT	((Atom)	66)
#define	XA_WM_CLASS	((Atom)	67)
#define	XA_WM_TRANSIENT_FOR	((Atom)	68)
#define	XA_LAST_PREDEFINED	((Atom)	68)

6-130 LIBRARIES



Figure 6-155: <X11/cursorfont.h>, Part 1 of 3

#define	XC_num_glyphs	154
#define	XC_X_cursor	0
#define	XC_arrow	2
#define	XC_based_arrow_down	4
#define	XC_based_arrow_up	6
#define	XC_boat	8
#define	XC_bogosity	10
#define	XC_bottom_left_corner	12
#define	XC_bottom_right_corner	14
#define	XC_bottom_side	16
#define	XC_bottom_tee	18
#define	XC_box_spiral	20
#define	XC_center_ptr	22
• •	XC_circle	24
	XC_clock	26
#define	XC_coffee_mug	28
#define	XC_cross	30
#define	XC_cross_reverse	32
	XC_crosshair	34
#define	XC_diamond_cross	36
#define	XC_dot	38
#define	XC_dotbox	40
	XC_double_arrow	42
#define	XC_draft_large	44
#define	XC_draft_small	46
	XC_draped_box	48
	XC_exchange	50
	XC_fleur	52
#define	XC_gobbler	54
	XC_gumby	56
**	XC_hand1	58
#define	XC_hand2	60

6-132 LIBRARIES

Figure 6-156: <X11/cursorfont.h>, Part 2 of 3

	#define	XC_heart	62
I	#define	XC_icon	64
I	#define	XC_iron_cross	66
I	#define	XC_left_ptr	68
I	#define	XC_left_side	70
I	#define	XC_left_tee	72
I	#define	XC_leftbutton	74
I	#define	XC_ll_angle	76
I	#define	XC_lr_angle	78
I	#define	XC_man	80
I	#define	XC_middlebutton	82
I	#define	XC_mouse	84
I	#define	XC_pencil	86
I	#define	XC_pirate	88
I	#define	XC_plus	90
I	#define	XC_question_arrow	92
I		XC_right_ptr	94
I		XC_right_side	96
I		XC_right_tee	98
I	#define	XC_rightbutton	100
I	#define	XC_rtl_logo	102
I	••	XC_sailboat	104
I		XC_sb_down_arrow	106
I		XC_sb_h_double_arrow	108
I		XC_sb_left_arrow	110
I		XC_sb_right_arrow	112
I		XC_sb_up_arrow	114
I	#define	XC_sb_v_double_arrow	116
		XC_shuttle	118
	#define	XC_sizing	120
	#define	XC_spider	122
	#define	XC_spraycan	124
I			

Figure 6-157: <X11/cursorfont.h>, Part 3 of 3

#define	XC_star	126
#define	XC_target	128
#define	XC_tcross	130
#define	XC_top_left_arrow	132
#define	XC_top_left_corner	134
#define	XC_top_right_corner	136
#define	XC_top_side	138
#define	XC_top_tee	140
#define	XC_trek	142
#define	XC_ul_angle	144
#define	XC_umbrella	146
#define	XC_ur_angle	148
#define	XC_watch	150
#define	XC_xterm	152

6-134 LIBRARIES

Figure 6-158: <X11/Intrinsic.h>, Part 1 of 6

```
typedef char *String;
#define XtNumber(arr)\
      ((Cardinal) (sizeof(arr) / sizeof(arr[0])))
typedef void
                           *Widget;
typedef Widget
                           *WidgetList;
typedef void
                           *CompositeWidget;
                           *WidgetClass;
typedef void
typedef XtActionsRec
                           *XtActionList;
typedef void
                           *XtAppContext;
typedef unsigned long
                          XtValueMask;
typedef unsigned long
                          XtIntervalId;
typedef unsigned long
                          XtInputId;
typedef unsigned long
                          XtWorkProcId;
typedef unsigned int
                          XtGeometryMask;
typedef unsigned long
                          XtGCMask;
typedef unsigned long
                          Pixel;
typedef int
                          XtCacheType;
#define
             XtCacheNone
                                 0x001
#define
                                 0x002
             XtCacheAll
#define
             XtCacheByDisplay
                                 0x003
#define
             XtCacheRefCount
                                 0x100
typedef char
                          Boolean;
typedef long
                          XtArqVal;
typedef unsigned char
                          XtEnum;
                          Cardinal;
typedef unsigned int
                          Dimension;
typedef unsigned short
typedef short
                          Position;
typedef void
                           *XtPointer;
```

Figure 6-159: <X11/Intrinsic.h>, Part 2 of 6

```
typedef void
                          *XtTranslations;
typedef void
                          *XtAccelerators;
typedef unsigned int
                          Modifiers;
#define XtCWQueryOnly
                          (1 << 7)
#define XtSMDontChange
typedef void *XtCacheRef;
typedef void *XtActionHookId;
typedef unsigned long
                          EventMask;
typedef enum {XtListHead, XtListTail } XtListPosition;
typedef unsigned long
                        XtInputMask;
typedef struct {
      String
                    string;
      XtActionProc proc;
} XtActionsRec;
typedef enum {
      XtAddress,
      XtBaseOffset,
      XtImmediate,
      XtResourceString,
      XtResourceQuark,
      XtWidgetBaseOffset,
      XtProcedureArg
} XtAddressMode;
typedef struct {
      XtAddressMode
                          address_mode;
      XtPointer
                          address_id;
      Cardinal
                          size;
} XtConvertArgRec, *XtConvertArgList;
```

6-136 LIBRARIES

Figure 6-160: <X11/Intrinsic.h>, Part 3 of 6

```
#define XtInputNoneMask
                                 0L
#define XtInputReadMask
                                  (1L << 0)
#define XtInputWriteMask
                                  (1L << 1)
#define XtInputExceptMask
                                  (1L << 2)
typedef struct {
      XtGeometryMask request_mode;
      Position
                    x, y;
      Dimension
                    width, height, border_width;
      Widget sibling;
} XtWidgetGeometry;
typedef struct {
      String
                    name;
      XtArgVal
                    value;
} Arg, *ArgList;
typedef XtPointer
                   XtVarArgsList;
typedef struct {
      XtCallbackProc
                           callback;
      XtPointer
                           closure;
} XtCallbackRec, *XtCallbackList;
typedef enum {
      XtCallbackNoList,
      XtCallbackHasNone,
      XtCallbackHasSome
} XtCallbackStatus;
typedef struct {
      Widget
                    shell_widget;
      Widget
                    enable_widget;
} XtPopdownIDRec, *XtPopdownID;
```

Figure 6-161: <X11/Intrinsic.h>, Part 4 of 6

```
typedef enum {
      XtGeometryYes,
      XtGeometryNo,
      XtGeometryAlmost,
      XtGeometryDone
} XtGeometryResult;
typedef enum {
      XtGrabNone,
      XtGrabNonexclusive,
      XtGrabExclusive
} XtGrabKind;
typedef struct {
      String
                   resource_name;
      String
                   resource_class;
      String
                   resource_type;
      Cardinal
                  resource_size;
      Cardinal
                  resource_offset;
      String
                   default_type;
                   default_addr;
      XtPointer
} XtResource, *XtResourceList;
typedef struct {
      char
                   match;
      String
                   substitution;
} SubstitutionRec, *Substitution;
typedef Boolean
                          (*XtFilePredicate);
typedef XtPointer
                          XtRequestId;
extern XtConvertArgRec const colorConvertArgs[];
extern XtConvertArgRec const screenConvertArg[];
```

6-138 LIBRARIES

Figure 6-162: <X11/Intrinsic.h>, Part 5 of 6

```
#define XtAllEvents ((EventMask) -1L)
#define XtIMXEvent
#define XtIMTimer
#define XtIMAlternateInput
                                    4
#define XtIMAll (XtIMXEvent | XtIMTimer | XtIMAlternateInput)
#define XtOffsetOf(s_type,field) XtOffset(s_type*,field)
#define XtNew(type) ((type *) XtMalloc((unsigned) sizeof(type)))
#define XT_CONVERT_FAIL
                                    (Atom)0x8000001
#define XtIsRectObj(object) \
       (_XtCheckSubclassFlag(object,(XtEnum)0x02))
#define XtIsWidget(object) \
       (_XtCheckSubclassFlag(object,(XtEnum)0x04))
#define XtIsComposite(widget) \
       (_XtCheckSubclassFlag(widget,(XtEnum)0x08))
#define XtIsConstraint(widget) \
       (_XtCheckSubclassFlag(widget,(XtEnum)0x10))
#define XtIsShell(widget) \
       (_XtCheckSubclassFlag(widget,(XtEnum)0x20))
#define XtIsOverrideShell(widget) \
       (_XtIsSubclassOf(widget,(WidgetClass)overrideShellWidgetClass,\
        (WidgetClass)shellWidgetClass, (XtEnum)0x20))
#define XtIsWMShell(widget) \
        (_XtCheckSubclassFlag(widget,(XtEnum)0x40))
#define XtIsVendorShell(widget)
        (_XtIsSubclassOf(widget,(WidgetClass)vendorShellWidgetClass,\
        (WidgetClass)wmShellWidgetClass,(XtEnum)0x40))
#define XtIsTransientShell(widget) \
        (_XtIsSubclassOf(widget,(WidgetClass)transientShellWidgetClass,\
        (WidgetClass)wmShellWidgetClass, (XtEnum)0x40))
#define XtIsTopLevelShell(widget)\
        (_XtCheckSubclassFlag(widget, (XtEnum)0x80))
#define XtIsApplicationShell(widget)\
        (_XtIsSubclassOf(widget,(WidgetClass)applicationShellWidgetClass,\
        (WidgetClass)topLevelShellWidgetClass, (XtEnum)0x80))
```

Figure 6-163: <X11/Intrinsic.h>, Part 6 of 6

```
#define XtSetArg(arg,n,d)\
      ((void)((arg).name = (n), (arg).value = (XtArgVal)(d)))
#define XtOffset(p_type,field)\
      #define XtVaNestedList
                               "XtVaNestedList"
#define XtVaTypedArg
                         "XtVaTypedArg"
#define XtUnspecifiedPixmap ((Pixmap)2)
#define XtUnspecifiedShellInt
                               (-1)
#define XtUnspecifiedWindow ((Window)2)
#define XtUnspecifiedWindowGroup ((Window)3)
#define XtDefaultForeground "XtDefaultForeground"
#define XtDefaultBackground "XtDefaultBackground"
#define XtDefaultFont
                         "XtDefaultFont"
#define XtDefaultFontSet
                               "XtDefaultFontSet"
```

Figure 6-164: <X11/Object.h>

extern WidgetClass objectClass;

Figure 6-165: <X11/RectObj.h>

extern WidgetClass rectObjClass;

6-140 LIBRARIES

Figure 6-166: <X11/extensions/shape.h>*

#define	ShapeSet	0
#define	ShapeUnion	1
#define	ShapeIntersect	2
#define	ShapeSubtract	3
#define	ShapeInvert	4
#define	ShapeBounding	0
#define	ShapeClip	1
	ShapeNotifyMask ShapeNotify	(1L << 0)

Figure 6-167: <X11/Shell.h>

```
extern WidgetClass shellWidgetClass;
extern WidgetClass overrideShellWidgetClass;
extern WidgetClass wmShellWidgetClass;
extern WidgetClass transientShellWidgetClass;
extern WidgetClass topLevelShellWidgetClass;
extern WidgetClass applicationShellWidgetClass;
```

Figure 6-168: <X11/Vendor.h>

extern WidgetClass vendorShellWidgetClass;

System Data Interfaces

Figure 6-169: <X11/X.h>, Part 1 of 12

```
typedef unsigned long XID;
typedef XID Window;
typedef XID Drawable;
typedef XID Font;
typedef XID Pixmap;
typedef XID Cursor;
typedef XID Colormap;
typedef XID GContext;
typedef XID KeySym;
typedef unsigned long Atom;
typedef unsigned long VisualID;
typedef unsigned long Time;
typedef unsigned char KeyCode;
#define AllTemporary
#define AnyButton
                          0L
#define AnyKey
                          0L
#define AnyPropertyType
                          0L
#define CopyFromParent
                          0L
#define CurrentTime
                          0L
#define InputFocus
                          1L
#define NoEventMask
                          0L
#define None
                          0L
#define NoSymbol
                          0L
#define ParentRelative
#define PointerWindow
                          ΟL
#define PointerRoot
```

6-142 LIBRARIES

Figure 6-170: <X11/X.h>, Part 2 of 12

J			
	#define	KeyPressMask	(1L<<0)
	#define	KeyReleaseMask	(1L<<1)
	#define	ButtonPressMask	(1L<<2)
	#define	ButtonReleaseMask	(1L<<3)
	#define	EnterWindowMask	(1L<<4)
	#define	LeaveWindowMask	(1L<<5)
	#define	PointerMotionMask	(1L<<6)
	#define	PointerMotionHintMask	(1L<<7)
	#define	Button1MotionMask	(1L<<8)
	#define	Button2MotionMask	(1L<<9)
	#define	Button3MotionMask	(1L<<10)
	#define	Button4MotionMask	(1L<<11)
	#define	Button5MotionMask	(1L<<12)
	#define	ButtonMotionMask	(1L<<13)
	#define	KeymapStateMask	(1L<<14)
	#define	ExposureMask	(1L<<15)
	#define	VisibilityChangeMask	(1L<<16)
	#define	StructureNotifyMask	(1L<<17)
	#define	ResizeRedirectMask	(1L<<18)
	#define	SubstructureNotifyMask	(1L<<19)
	#define	${\tt Substructure Redirect Mask}$	(1L<<20)
	#define	FocusChangeMask	(1L<<21)
	#define	PropertyChangeMask	(1L<<22)
	#define	ColormapChangeMask	(1L<<23)
	#define	OwnerGrabButtonMask	(1L<<24)

Figure 6-171: <X11/X.h>, Part 3 of 12

#define	KeyPress	2
#define	KeyRelease	3
#define	ButtonPress	4
#define	ButtonRelease	5
#define	MotionNotify	6
#define	EnterNotify	7
#define	LeaveNotify	8
#define	FocusIn	9
#define	FocusOut	10
#define	KeymapNotify	11
#define	Expose	12
#define	GraphicsExpose	13
#define	NoExpose	14
#define	VisibilityNotify	15
#define	CreateNotify	16
#define	DestroyNotify	17
#define	UnmapNotify	18
#define	MapNotify	19
	MapRequest	20
	ReparentNotify	21
	ConfigureNotify	22
#define	ConfigureRequest	23
#define	GravityNotify	24
#define	ResizeRequest	25
	CirculateNotify	26
#define	CirculateRequest	27
1	PropertyNotify	28
	SelectionClear	29
	SelectionRequest	30
	SelectionNotify	31
	ColormapNotify	32
1	ClientMessage	33
	MappingNotify	34
#define	LASTEvent	35
	/* must be bigger than	any event # */

6-144 LIBRARIES

Figure 6-172: <X11/X.h>, Part 4 of 12

#define	ShiftMask	(1<<0)
#define	LockMask	(1<<1)
#define	ControlMask	(1<<2)
#define	Mod1Mask	(1<<3)
#define	Mod2Mask	(1<<4)
#define	Mod3Mask	(1<<5)
#define	Mod4Mask	(1<<6)
#define	Mod5Mask	(1<<7)
#define	Button1Mask	(1<<8)
	Button2Mask	(1<<9)
	Button3Mask	(1<<10)
	Button4Mask	(1<<11)
	Button5Mask	(1<<12)
#define	AnyModifier	(1<<15)
#define	Button1	1
#define	Button2	2
	Button3	3
	Button4	4
#define	Button5	5
	NotifyNormal	0
	NotifyGrab	1
	NotifyUngrab	2
	NotifyWhileGrabbed	
	NotifyHint	1
	NotifyAncestor	0
	NotifyVirtual	1
	NotifyInferior	2
	NotifyNonlinear	3
	NotifyNonlinearVir	
	NotifyPointer	5
	NotifyPointerRoot	6
#define	NotifyDetailNone	7
1		

Figure 6-173: <X11/X.h>, Part 5 of 12

#define	VisibilityUnobscured VisibilityPartiallyObscu	
#define	VisibilityFullyObscured	2
	PlaceOnTop PlaceOnBottom	0
#define	PropertyNewValue	0
#define	PropertyDelete	1
#define	ColormapUninstalled	0
#define	ColormapInstalled	1
#define	GrabModeSync	0
#define	GrabModeAsync	1
1	GrabSuccess	0
	AlreadyGrabbed	1
	GrabInvalidTime	2
	GrabNotViewable	3
#define	GrabFrozen	4
#define	AsyncPointer	0
#define	SyncPointer	1
#define	ReplayPointer	2
#define	AsyncKeyboard	3
#define	SyncKeyboard	4
#define	ReplayKeyboard	5
	AsyncBoth	6
#define	SyncBoth	7
	RevertToNone	(int)None
	RevertToPointerRoot	(int)PointerRoot
#define	RevertToParent	2

6-146 LIBRARIES

Figure 6-174: <X11/X.h>, Part 6 of 12

#define	Success	0
#define	BadRequest	1
	BadValue	2
#define	BadWindow	3
#define	BadPixmap	4
	BadAtom	5
#define	BadCursor	6
#define	BadFont	7
#define	BadMatch	8
#define	BadDrawable	9
#define	BadAccess	10
	BadAlloc	11
	BadColor	12
#define	BadGC	13
	BadIDChoice	14
	BadName	15
	BadLength	16
#define	BadImplementation	17
#define	InputOutput	1
#define	InputOnly	2
	CWBackPixmap	(1L<<0)
	CWBackPixel	(1L<<1)
#define	CWBorderPixmap	(1L<<2)
#define	CWBorderPixel	(1L<<3)
#define	CWBitGravity	(1L<<4)
	CWWinGravity	(1L<<5)
#define	CWBackingStore	(1L<<6)
	CWBackingPlanes	(1L<<7)
	CWBackingPixel	(1L<<8)
#define	CWOverrideRedirect	
	CWSaveUnder	(1L<<10)
	CWEventMask	(1L<<11)
#define	CWDontPropagate	(1L<<12)
#define	CWColormap	(1L<<13)
#define	CWCursor	(1L<<14)

Figure 6-175: <X11/X.h>, Part 7 of 12

#define	CWX	(1<<0)	
#define	CWY	(1<<1)	
#define	CWWidth	(1<<2)	
#define	CWHeight	(1<<3)	
#define	CWBorderWidth	(1 << 4)	
#define	CWSibling	(1<<5)	
#define	CWStackMode	(1<<6)	
#define	ForgetGravity		0
#define	NorthWestGravity		1
#define	NorthGravity		2
#define	NorthEastGravity		3
#define	WestGravity		4
	CenterGravity		5
	EastGravity		6
#define	SouthWestGravity		7
#define	SouthGravity		8
**	SouthEastGravity		9
	StaticGravity		10
#define	UnmapGravity		0
	NotUseful		0
	WhenMapped		1
#define	Always		2
	1		•
	IsUnmapped		0
	IsUnviewable		1
#define	IsViewable		2
ша-е:	SetModeInsert		0
	SetModeInsert		0
#deline	sermodebetere		Т
#define	DestroyAll		0
	RetainPermanent		1
	RetainTemporary		2
HACTING	1.coaliii ciipotat y		_

6-148 LIBRARIES

Figure 6-176: <X11/X.h>, Part 8 of 12

#define	Above	0
#define	Below	1
#define	TopIf	2
#define	BottomIf	3
#define	Opposite	4
#define	RaiseLowest	0
#define	LowerHighest	1
#define	PropModeReplace	0
#define	PropModePrepend	1
#define	PropModeAppend	2
	GXclear	0×0
#define		0x1
	GXandReverse	0x2
#define		0x3
	GXandInverted	0x4
#define	GXnoop	0x5
#define		0x6
#define	GXor	0x7
#define		8x0
#define	GXequiv	0x9
#define	GXinvert	0xa
#define	GXorReverse	dx0
#define	GXcopyInverted	0xc
#define	GXorInverted	0xd
#define		0xe
#define	GXset	0xf
	LineSolid	0
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	LineOnOffDash	1
	LineDoubleDash	2
	CapNotLast	0
#define	-	1
#define		2
#define	CapProjecting	3
·		

Figure 6-177: <X11/X.h>, Part 9 of 12

#define	JoinMiter	0
#define	JoinRound	1
#define	JoinBevel	2
#define	FillSolid	0
#define	FillTiled	1
#define	FillStippled	2
#define	FillOpaqueStippled	.3
#define	EvenOddRule	0
#define	WindingRule	1
#define	ClipByChildren	0
#define	IncludeInferiors	1
#define	Unsorted	0
#define	YSorted	1
#define	YXSorted	2
#define	YXBanded	3
#define	CoordModeOrigin	0
#define	CoordModePrevious	1
	Complex	0
	Nonconvex	1
#define	Convex	2
	ArcChord	0
#define	ArcPieSlice	1

6-150 LIBRARIES

Figure 6-178: <X11/X.h>, Part 10 of 12

#define	GCFunction	(1L<<0)
#define	GCPlaneMask	(1L<<1)
#define	GCForeground	(1L<<2)
#define	GCBackground	(1L<<3)
#define	GCLineWidth	(1L<<4)
#define	GCLineStyle	(1L<<5)
#define	GCCapStyle	(1L<<6)
	GCJoinStyle	(1L<<7)
#define	GCFillStyle	(1L<<8)
#define	GCFillRule	(1L<<9)
#define	GCTile	(1L<<10)
#define	GCStipple	(1L<<11)
#define	GCTileStipXOrigin	(1L<<12)
#define	GCTileStipYOrigin	(1L<<13)
#define	GCFont	(1L<<14)
#define	GCSubwindowMode	(1L<<15)
	GCGraphicsExposures	(1L<<16)
	GCClipXOrigin	(1L<<17)
	GCClipYOrigin	(1L<<18)
#define	GCClipMask	(1L<<19)
	GCDashOffset	(1L<<20)
••	GCDashList	(1L<<21)
#define	GCArcMode	(1L<<22)
		_
	FontLeftToRight	0
#define	FontRightToLeft	1
#define	XYBitmap	0
	XYPixmap	1
	ZPixmap	2
.,		
#define	AllocNone	0
#define	AllocAll	1
#define	DoRed	(1<<0)
	DoGreen	(1<<1)
#define	DoBlue	(1<<2)

Figure 6-179: <X11/X.h>, Part 11 of 12

#define	CursorShape	0
#define	TileShape	1
#define	StippleShape	2
#define	AutoRepeatModeOff	0
#define	AutoRepeatModeOn	1
#define	AutoRepeatModeDefault	2
,,	LedModeOff	0
#define	LedModeOn	1
11 3 - 6 2	VDV	(15 + 40)
	KBKeyClickPercent KBBellPercent	(1L<<0)
		(1L<<1)
	KBBellPitch	(1L<<2)
	KBBellDuration	(1L<<3)
#define		(1L<<4)
	KBLedMode	(1L<<5)
#define	-	(1L<<6)
#define	KBAutoRepeatMode	(1L<<7)
#dofino	MappingSuccess	0
	MappingBusy	1
	MappingFailed	2
#deline	Mappingraffed	2
#define	MappingModifier	0
#define	MappingKeyboard	1
#define	MappingPointer	2
#define	DontPreferBlanking	0
#define	PreferBlanking	1
#define	DefaultBlanking	2
	DontAllowExposures	0
	AllowExposures	1
#define	DefaultExposures	2

6-152 LIBRARIES

Figure 6-180: <X11/X.h>, Part 12 of 12

	ScreenSaverReset ScreenSaverActive	0
#define #define	EnableAccess DisableAccess StaticGray GrayScale	1 0 0 1
#define #define	StaticColor PseudoColor TrueColor DirectColor	2 3 4 5
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	LSBFirst MSBFirst	0 1

Figure 6-181: <X11/Xcms.h>, Part 1 of 5

```
#define XcmsFailure
                                       0
#define XcmsSuccess
                                       1
#define XcmsSuccessWithCompression
#define XcmsUndefinedFormat
                                 (XcmsColorFormat)0x00000000
#define XcmsCIEXYZFormat
                                 (XcmsColorFormat)0x0000001
#define XcmsCIEuvYFormat
                                 (XcmsColorFormat)0x0000002
#define XcmsCIExyYFormat
                                 (XcmsColorFormat)0x0000003
#define XcmsCIELabFormat
                                 (XcmsColorFormat)0x0000004
#define XcmsCIELuvFormat
                                 (XcmsColorFormat)0x0000005
#define XcmsTekHVCFormat
                                 (XcmsColorFormat)0x0000006
#define XcmsRGBFormat
                                 (XcmsColorFormat)0x80000000
#define XcmsRGBiFormat
                                 (XcmsColorFormat)0x80000001
#define XcmsInitNone
                                 0x00
#define XcmsInitSuccess
                                 0x01
typedef unsigned int XcmsColorFormat;
typedef double XcmsFloat;
typedef struct {
      unsigned short red;
      unsigned short green;
      unsigned short blue;
} XcmsRGB;
```

6-154 LIBRARIES

Figure 6-182: <X11/Xcms.h>, Part 2 of 5

```
typedef struct {
      XcmsFloat red;
      XcmsFloat green;
      XcmsFloat blue;
} XcmsRGBi;
typedef struct {
      XcmsFloat X;
      XcmsFloat Y;
      XcmsFloat Z;
} XcmsCIEXYZ;
typedef struct {
      XcmsFloat u_prime;
      XcmsFloat v_prime;
      XcmsFloat Y;
} XcmsCIEuvY;
typedef struct {
      XcmsFloat x;
      XcmsFloat y;
      XcmsFloat Y;
} XcmsCIExyY;
typedef struct {
      XcmsFloat L_star;
      XcmsFloat a_star;
      XcmsFloat b_star;
} XcmsCIELab;
```

Figure 6-183: <X11/Xcms.h>, Part 3 of 5

```
typedef struct {
      XcmsFloat L_star;
      XcmsFloat u_star;
      XcmsFloat v_star;
} XcmsCIELuv;
typedef struct {
      XcmsFloat H;
      XcmsFloat V;
      XcmsFloat C;
} XcmsTekHVC;
typedef struct {
      XcmsFloat pad0;
      XcmsFloat pad1;
      XcmsFloat pad2;
      XcmsFloat pad3;
} XcmsPad;
```

6-156 LIBRARIES

Figure 6-184: <X11/Xcms.h>, Part 4 of 5

```
typedef struct {
      union {
             XcmsRGB
                          RGB;
             XcmsRGBi
                          RGBi;
             XcmsCIEXYZ
                          CIEXYZ;
             XcmsCIEuvY
                          CIEuvY;
             XcmsCIExyY
                          CIExyY;
             XcmsCIELab
                          CIELab;
             XcmsCIELuv
                          CIELuv;
             XcmsTekHVC
                          TekHVC;
             XcmsPad
                          Pad;
             spec;
      unsigned longpixel;
      XcmsColorFormat
                          format;
} XcmsColor;
typedef struct {
      XcmsColor
                          screenWhitePt;
      XPointer
                          functionSet;
      XPointer
                          screenData;
      unsigned char
                          state;
      char
                          pad[3];
} XcmsPerScrnInfo;
typedef void *XcmsCCC;
typedef Status (*XcmsConversionProc)();
typedef XcmsConversionProc *XcmsFuncListPtr;
```

Figure 6-185: <X11/Xcms.h>, Part 5 of 5

```
typedef struct {
      char
                                 *prefix;
      XcmsColorFormat
                                 id;
      XcmsParseStringProc
                                 parseString;
      XcmsFuncListPtr
                                 to_CIEXYZ;
      XcmsFuncListPtr
                                 from_CIEXYZ;
      int
                                 inverse_flag;
} XcmsColorSpace;
typedef struct {
      XcmsColorSpace
                                 **DDColorSpaces;
      XcmsScreenInitProc
                                 screenInitProc;
      XcmsScreenFreeProc
                                 screenFreeProc;
} XcmsFunctionSet;
```

6-158 LIBRARIES

Figure 6-186: <X11/Xlib.h>, Part 1 of 27

```
typedef void *XPointer;
#define Bool
                                 int
#define Status
                                 int
#define True
                                 1
#define False
                                 0
#define QueuedAlready
                                 0
#define QueuedAfterReading
                                 1
#define QueuedAfterFlush
#define AllPlanes
                                 ((unsigned long)~0L)
```

Figure 6-187: <X11/Xlib.h>, Part 2 of 27

```
typdef void XExtData;

typdef void XExtCodes;

typedef struct {
    int depth;
    int bits_per_pixel;
    int scanline_pad;
} XPixmapFormatValues;
```

Figure 6-188: <X11/Xlib.h>, Part 3 of 27

```
typedef struct {
      int function;
      unsigned long plane_mask;
      unsigned long foreground;
      unsigned long background;
      int line_width;
      int line_style;
      int cap_style;
      int join_style;
      int fill_style;
      int fill_rule;
      int arc mode;
      Pixmap tile;
      Pixmap stipple;
      int ts_x_origin;
      int ts_y_origin;
      Font font;
      int subwindow_mode;
      Bool graphics_exposures;
      int clip_x_origin;
      int clip_y_origin;
      Pixmap clip_mask;
      int dash_offset;
      char dashes;
} XGCValues;
typedef void *GC;
typedef struct _dummy Visual;
```

6-160 LIBRARIES

Figure 6-189: <X11/Xlib.h>, Part 4 of 27

```
typedef struct _dummy Screen;
typedef struct {
      Pixmap background_pixmap;
      unsigned long background_pixel;
      Pixmap border_pixmap;
      unsigned long border_pixel;
      int bit_gravity;
      int win_gravity;
      int backing_store;
      unsigned long backing_planes;
      unsigned long backing_pixel;
      Bool save_under;
      long event_mask;
      long do_not_propagate_mask;
      Bool override_redirect;
      Colormap colormap;
      Cursor cursor;
} XSetWindowAttributes;
```

Figure 6-190: <X11/Xlib.h>, Part 5 of 27

```
typedef struct _dummy ScreenFormat;
typedef struct {
      int x, y;
      int width, height;
      int border_width;
      int depth;
      Visual *visual;
      Window root;
      int class;
      int bit_gravity;
      int win_gravity;
      int backing_store;
      unsigned long backing_planes;
      unsigned long backing_pixel;
      Bool save_under;
      Colormap colormap;
      Bool map_installed;
      int map_state;
      long all_event_masks;
      long your_event_mask;
      long do_not_propagate_mask;
      Bool override redirect;
      Screen *screen;
} XWindowAttributes;
```

6-162 LIBRARIES

Figure 6-191: <X11/Xlib.h>, Part 6 of 27

```
typedef struct {
      int family;
      int length;
      char *address;
} XHostAddress;
typedef struct _XImage {
      int width, height;
      int xoffset;
      int format;
      char *data;
      int byte order;
      int bitmap_unit;
      int bitmap_bit_order;
      int bitmap_pad;
      int depth;
      int bytes_per_line;
      int bits_per_pixel;
      unsigned long red_mask;
      unsigned long green_mask;
      unsigned long blue_mask;
      XPointer obdata;
      struct funcs {
             struct _XImage *(*create_image)();
             int (*destroy_image)();
             unsigned long (*get_pixel)();
             int (*put_pixel)();
             struct _XImage *(*sub_image)();
             int (*add_pixel)();
      } f;
} XImage;
```

Figure 6-192: <X11/Xlib.h>, Part 7 of 27

```
typedef struct {
      int x, y;
      int width, height;
      int border_width;
      Window sibling;
      int stack_mode;
} XWindowChanges;
typedef struct {
      unsigned long pixel;
      unsigned short red, green, blue;
      char flags;
      char pad;
} XColor;
typedef struct {
      short x1, y1, x2, y2;
} XSegment;
typedef struct {
      short x, y;
} XPoint;
typedef struct {
      short x, y;
      unsigned short width, height;
} XRectangle;
typedef struct {
      short x, y;
      unsigned short width, height;
      short angle1, angle2;
} XArc;
```

6-164 LIBRARIES

Figure 6-193: <X11/Xlib.h>, Part 8 of 27

```
typedef struct {
      int key_click_percent;
      int bell_percent;
      int bell_pitch;
      int bell_duration;
      int led;
      int led_mode;
      int key;
      int auto_repeat_mode;
} XKeyboardControl;
typedef struct {
      int key_click_percent;
      int bell_percent;
      unsigned int bell_pitch, bell_duration;
      unsigned long led_mask;
      int global_auto_repeat;
      char auto_repeats[32];
} XKeyboardState;
typedef struct {
      Time time;
      short x, y;
} XTimeCoord;
typedef struct {
      int
                   max_keypermod;
      KeyCode
                   *modifiermap;
} XModifierKeymap;
typedef struct _dummy Display;
```

Figure 6-194: <X11/Xlib.h>, Part 9 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Window root;
      Window subwindow;
      Time time;
      int x, y;
      int x_root, y_root;
      unsigned int state;
      unsigned int keycode;
      Bool same screen;
} XKeyEvent;
typedef XKeyEvent XKeyPressedEvent;
typedef XKeyEvent XKeyReleasedEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Window root;
      Window subwindow;
      Time time;
      int x, y;
      int x_root, y_root;
      unsigned int state;
      unsigned int button;
      Bool same_screen;
} XButtonEvent;
typedef XButtonEvent XButtonPressedEvent;
typedef XButtonEvent XButtonReleasedEvent;
```

6-166 LIBRARIES

Figure 6-195: <X11/Xlib.h>, Part 10 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Window root;
      Window subwindow;
      Time time;
      int x, y;
      int x_root, y_root;
      unsigned int state;
      char is_hint;
      Bool same_screen;
} XMotionEvent;
typedef XMotionEvent XPointerMovedEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Window root;
      Window subwindow;
      Time time;
      int x, y;
      int x_root, y_root;
      int mode;
      int detail;
      Bool same_screen;
      Bool focus;
      unsigned int state;
} XCrossingEvent;
```

Figure 6-196: <X11/Xlib.h>, Part 11 of 27

```
typedef XCrossingEvent XEnterWindowEvent;
typedef XCrossingEvent XLeaveWindowEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      int mode;
      int detail;
} XFocusChangeEvent;
typedef XFocusChangeEvent XFocusInEvent;
typedef XFocusChangeEvent XFocusOutEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      char key_vector[32];
} XKeymapEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      int x, y;
      int width, height;
      int count;
} XExposeEvent;
```

6-168 LIBRARIES

Figure 6-197: <X11/Xlib.h>, Part 12 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Drawable drawable;
      int x, y;
      int width, height;
      int count;
      int major_code;
      int minor_code;
} XGraphicsExposeEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Drawable drawable;
      int major_code;
      int minor_code;
} XNoExposeEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      int state;
} XVisibilityEvent;
```

Figure 6-198: <X11/Xlib.h>, Part 13 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window parent;
      Window window;
      int x, y;
      int width, height;
      int border_width;
      Bool override_redirect;
} XCreateWindowEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window event;
      Window window;
} XDestroyWindowEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window event;
      Window window;
      Bool from_configure;
} XUnmapEvent;
```

6-170 LIBRARIES

Figure 6-199: <X11/Xlib.h>, Part 14 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window event;
      Window window;
      Bool override_redirect;
} XMapEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window parent;
      Window window;
} XMapRequestEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send event;
      Display *display;
      Window event;
      Window window;
      Window parent;
      int x, y;
      Bool override_redirect;
} XReparentEvent;
```

Figure 6-200: <X11/Xlib.h>, Part 15 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window event;
      Window window;
      int x, y;
      int width, height;
      int border_width;
      Window above;
      Bool override_redirect;
} XConfigureEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window event;
      Window window;
      int x, y;
} XGravityEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      int width, height;
} XResizeRequestEvent;
```

6-172 LIBRARIES

Figure 6-201: <X11/Xlib.h>, Part 16 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window parent;
      Window window;
      int x, y;
      int width, height;
      int border_width;
      Window above;
      int detail;
      unsigned long value_mask;
} XConfigureRequestEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window event;
      Window window;
      int place;
} XCirculateEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window parent;
      Window window;
      int place;
} XCirculateRequestEvent;
```

Figure 6-202: <X11/Xlib.h>, Part 17 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Atom atom;
      Time time;
      int state;
} XPropertyEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Atom selection;
      Time time;
} XSelectionClearEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window owner;
      Window requestor;
      Atom selection;
      Atom target;
      Atom property;
      Time time;
} XSelectionRequestEvent;
```

6-174 LIBRARIES

Figure 6-203: <X11/Xlib.h>, Part 18 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window requestor;
      Atom selection;
      Atom target;
      Atom property;
      Time time;
} XSelectionEvent;
typedef struct {
      int type;
      Display *display;
      XID resourceid;
      unsigned long serial;
      unsigned char error_code;
      unsigned char request_code;
      unsigned char minor_code;
} XErrorEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Atom message_type;
      int format;
      union {
             char b[20];
             short s[10];
             long 1[5];
      } data;
} XClientMessageEvent;
```

Figure 6-204: <X11/Xlib.h>, Part 19 of 27

```
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      Colormap colormap;
      Bool new;
      int state;
} XColormapEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
      int request;
      int first_keycode;
      int count;
} XMappingEvent;
typedef struct {
      int type;
      unsigned long serial;
      Bool send_event;
      Display *display;
      Window window;
} XAnyEvent;
```

6-176 LIBRARIES

Figure 6-205: <X11/Xlib.h>, Part 20 of 27

```
typedef union _XEvent {
      int
                                  type;
      XAnyEvent
                                  xany;
      XKeyEvent
                                  xkey;
      XButtonEvent
                                  xbutton;
      XMotionEvent
                                  xmotion;
      XCrossingEvent
                                  xcrossing;
      XFocusChangeEvent
                                  xfocus;
      XExposeEvent
                                  xexpose;
      XGraphicsExposeEvent
                                  xgraphicsexpose;
      XNoExposeEvent
                                  xnoexpose;
      XVisibilityEvent
                                  xvisibility;
      XCreateWindowEvent
                                  xcreatewindow;
      XDestroyWindowEvent
                                  xdestroywindow;
      XUnmapEvent
                                  xunmap;
      XMapEvent
                                  xmap;
      XMapRequestEvent
                                  xmaprequest;
      XReparentEvent
                                  xreparent;
      XConfigureEvent
                                  xconfigure;
      XGravityEvent
                                  xgravity;
                                  xresizerequest;
      XResizeRequestEvent
      XConfigureRequestEvent
                                  xconfigurerequest;
      XCirculateEvent
                                  xcirculate;
      XCirculateRequestEvent
                                  xcirculaterequest;
      XPropertyEvent
                                  xproperty;
      XSelectionClearEvent
                                  xselectionclear;
      XSelectionRequestEvent
                                  xselectionrequest;
      XSelectionEvent
                                  xselection;
      XColormapEvent
                                  xcolormap;
      XClientMessageEvent
                                  xclient;
      XMappingEvent
                                  xmapping;
      XErrorEvent
                                  xerror;
      XKeymapEvent
                                  xkeymap;
      long
                                 pad[24];
} XEvent;
```

Figure 6-206: <X11/Xlib.h>, Part 21 of 27

```
typedef struct {
      short lbearing;
      short rbearing;
      short width;
      short ascent;
      short descent;
      unsigned short attributes;
} XCharStruct;
typedef struct {
      Atom name;
      unsigned long card32;
} XFontProp;
typedef struct {
      XExtData
                    *ext_data;
      Font
                    fid;
      unsigned
                   direction;
      unsigned
                   min_char_or_byte2;
      unsigned
                   max_char_or_byte2;
      unsigned
                   min_byte1;
      unsigned
                   max_byte1;
                   all_chars_exist;
      Bool
                   default_char;
      unsigned
      int
                   n_properties;
      XFontProp
                   *properties;
      XCharStruct min_bounds;
      XCharStruct max_bounds;
      XCharStruct *per_char;
      int
                   ascent;
      int
                   descent;
} XFontStruct;
```

6-178 LIBRARIES

Figure 6-207: <X11/Xlib.h>, Part 22 of 27

```
typedef struct {
      char *chars;
      int nchars;
      int delta;
      Font font;
} XTextItem;
typedef struct {
      unsigned char bytel;
      unsigned char byte2;
} XChar2b;
typedef struct {
      XChar2b *chars;
      int nchars;
      int delta;
      Font font;
} XTextItem16;
typedef union {
      Display *display;
      GC gc;
      Visual *visual;
      Screen *screen;
      ScreenFormat *pixmap_format;
      XFontStruct *font;
} XEDataObject;
typedef struct {
      XRectangle
                   max_ink_extent;
      XRectangle
                   max_logical_extent;
} XFontSetExtents;
typedef struct _dummy XFontSet;
```

Figure 6-208: <X11/Xlib.h>, Part 23 of 27

```
typedef struct {
      char
                    *chars;
       int
                    nchars;
       int
                    delta;
      XFontSet
                    *font_set;
} XmbTextItem;
typedef struct {
      wchar_t
                    *chars;
       int
                    nchars;
                    delta;
       int
      XFontSet
                    font set;
} XwcTextItem;
typedef void (*XIMProc)();
typedef void *XIM;
typedef void *XIC;
typedef unsigned long XIMStyle;
typedef struct {
      unsigned short count_styles;
      XIMStyle *supported_styles;
} XIMStyles;
#define XIMPreeditArea
                                  0x0001L
#define XIMPreeditCallbacks
                                  0x0002L
#define XIMPreeditPosition
                                  0 \times 0004 L
#define XIMPreeditNothing
                                  0x0008L
#define XIMPreeditNone
                                  0x0010L
#define XIMStatusArea
                                  0 \times 0100 L
#define XIMStatusCallbacks
                                  0x0200L
#define XIMStatusNothing
                                  0x0400L
#define XIMStatusNone
                                  0x0800L
```

6-180 LIBRARIES

Figure 6-209: <X11/Xlib.h>, Part 24 of 27

#define	XNVaNestedList	"XNVaNestedList"
#define	XNQueryInputStyle	"queryInputStyle"
#define	XNClientWindow	"clientWindow"
#define	XNInputStyle	"inputStyle"
#define	XNFocusWindow	"focusWindow"
#define	XNResourceName	"resourceName"
#define	XNResourceClass	"resourceClass"
#define	XNGeometryCallback	"geometryCallback"
#define	XNFilterEvents	"filterEvents"
#define	XNPreeditStartCallback	"preeditStartCallback"
#define	XNPreeditDoneCallback	"preeditDoneCallback"
#define	XNPreeditDrawCallback	"preeditDrawCallback"
#define	XNPreeditCaretCallback	"preeditCaretCallback"
#define	XNPreeditAttributes	"preeditAttributes"
#define	XNStatusStartCallback	"statusStartCallback"
#define	XNStatusDoneCallback	"statusDoneCallback"
#define	XNStatusDrawCallback	"statusDrawCallback"
#define	XNStatusAttributes	"statusAttributes"
#define	XNArea	"area"
#define	XNAreaNeeded	"areaNeeded"
#define	XNSpotLocation	"spotLocation"
#define	XNColormap	"colorMap"
#define	XNStdColormap	"stdColorMap"
#define	XNForeground	"foreground"
#define	XNBackground	"background"
#define	XNBackgroundPixmap	"backgroundPixmap"
#define	XNFontSet	"fontSet"
#define	XNLineSpace	"lineSpace"
#define	XNCursor	"cursor"

Figure 6-210: <X11/Xlib.h>, Part 25 of 27

```
#define XBufferOverflow
                           -1
#define XLookupNone
                           1
#define XLookupChars
                           2
#define XLookupKeySym
                           3
#define XLookupBoth
typedef XPointer XVaNestedList;
typedef struct {
      XPointer client_data;
      XIMProc callback;
} XIMCallback;
typedef unsigned long XIMFeedback;
#define XIMReverse
#define XIMUnderline
                           (1 << 1)
#define XIMHighlight
                           (1 << 2)
#define XIMPrimary
                           (1 << 5)
#define XIMSecondary
                           (1 << 6)
#define XIMTertiary
                           (1 << 7)
typedef struct _XIMText {
      unsigned short length;
      XIMFeedback *feedback;
      Bool encoding_is_wchar;
      union {
             char *multi_byte;
             wchar_t *wide_char;
      } string;
} XIMText;
```

6-182 LIBRARIES

Figure 6-211: <X11/Xlib.h>, Part 26 of 27

```
typedef struct _XIMPreeditDrawCallbackStruct {
      int caret;
      int chg_first;
      int chg_length;
      XIMText *text;
} XIMPreeditDrawCallbackStruct;
typedef enum {
      XIMForwardChar, XIMBackwardChar,
      XIMForwardWord, XIMBackwardWord,
      XIMCaretUp, XIMCaretDown,
      XIMNextLine, XIMPreviousLine,
      XIMLineStart, XIMLineEnd,
      XIMAbsolutePosition,
      XIMDontChange
} XIMCaretDirection;
typedef enum {
      XIMIsInvisible,
      XIMIsPrimary,
      XIMIsSecondary
} XIMCaretStyle;
typedef struct _XIMPreeditCaretCallbackStruct {
      int position;
      XIMCaretDirection direction;
      XIMCaretStyle style;
} XIMPreeditCaretCallbackStruct;
```

Figure 6-212: <X11/Xlib.h>, Part 27 of 27

```
typedef enum {
     XIMTextType,
     XIMBitmapType
} XIMStatusDataType;

typedef struct _XIMStatusDrawCallbackStruct {
     XIMStatusDataType type;
     union {
          XIMText *text;
          Pixmap bitmap;
     } data;
} XIMStatusDrawCallbackStruct;
```

6-184 LIBRARIES

Figure 6-213: <X11/Xresource.h>, Part 1 of 2

```
typedef int
                     XrmQuark, *XrmQuarkList;
#define NULLQUARK
                      ((XrmQuark) 0)
typedef enum {XrmBindTightly, XrmBindLoosely} \
              XrmBinding, *XrmBindingList;
typedef XrmQuark
                                    XrmName;
typedef XrmQuarkList
                                    XrmNameList;
typedef XrmQuark
                                    XrmClass;
typedef XrmQuarkList
                                    XrmClassList;
typedef XrmQuark
                                    XrmRepresentation;
#define XrmStringToName(string)
                                           XrmStringToQuark(string)
#define XrmStringToNameList(str, name)
                                           XrmStringToQuarkList(str, name)
#define XrmStringToClass(class)
                                           XrmStringToQuark(class)
#define XrmStringToClassList(str,class)
                                           XrmStringToQuarkList(str, class)
#define XrmStringToRepresentation(string) XrmStringToQuark(string)
typedef struct {
       unsigned int
                             size;
       XPointer
                             addr;
} XrmValue, *XrmValuePtr;
typedef void
                              *XrmHashBucket;
typedef XrmHashBucket
                              *XrmHashTable;
typedef XrmHashTable
                              XrmSearchList[];
typedef void
                              *XrmDatabase;
#define XrmEnumAllLevels
                                     0
#define XrmEnumOneLevel
                                     1
```

Figure 6-214: <X11/Xresource.h>, Part 2 of 2

```
typedef enum {
      XrmoptionNoArg,
      XrmoptionIsArg,
      XrmoptionStickyArg,
      XrmoptionSepArg,
      XrmoptionResArg,
      XrmoptionSkipArg,
      XrmoptionSkipLine,
      XrmoptionSkipNArgs
} XrmOptionKind;
typedef struct {
      char
                           *option;
      char
                           *specifier;
      XrmOptionKind
                          argKind;
      XPointer
} XrmOptionDescRec, *XrmOptionDescList;
```

6-186 LIBRARIES

Figure 6-215: <X11/Xutil.h>, Part 1 of 5

```
#define NoValue
                    0x0000
#define XValue 0x0001
#define YValue 0x0002
#define WidthValue
                    0x0004
#define HeightValue 0x0008
#define AllValues
                    0x000F
#define XNegative
                    0 \times 0010
#define YNegative
                    0x0020
typedef struct {
      long flags;
      int x, y;
      int width, height;
      int min_width, min_height;
      int max_width, max_height;
      int width_inc, height_inc;
      struct {
             int x;
             int y;
      } min_aspect, max_aspect;
      int base_width, base_height;
      int win_gravity;
} XSizeHints;
#define USPosition
                    (1L << 0)
#define USSize(1L << 1)</pre>
#define PPosition (1L << 2)
#define PSize
                  (1L << 3)
                  (1L << 4)
#define PMinSize
#define PMaxSize
                  (1L << 5)
#define PResizeInc (1L << 6)</pre>
#define PAspect
                    (1L << 7)
#define PBaseSize
                    (1L << 8)
#define PWinGravity (1L << 9)</pre>
```

Figure 6-216: <X11/Xutil.h>, Part 2 of 5

```
typedefstruct {
       long
              flags;
       Bool
              input;
       int
              initial_state;
       Pixmap icon_pixmap;
       Window icon_window;
       int
              icon_x, icon_y;
       Pixmap icon_mask;
       XID
              window_group;
} XWMHints;
#define InputHint
                             (1L << 0)
                             (1L << 1)
#define StateHint
#define IconPixmapHint
                             (1L << 2)
#define IconWindowHint
                             (1L << 3)
#define IconPositionHint
                             (1L << 4)
#define IconMaskHint (1L << 5)</pre>
#define WindowGroupHint
                             (1L << 6)
#define AllHints (InputHint|StateHint|
       IconPixmapHint|IconWindowHint|
       IconPositionHint|IconMaskHint|WindowGroupHint)
#define WithdrawnState
#define NormalState
                             1
#define IconicState
typedef struct {
       unsigned char
                             *value;
       Atom
                             encoding;
       int
                             format;
       unsigned long
                             nitems;
} XTextProperty;
#define XNoMemory
                                     -1
#define XLocaleNotSupported -2
#define XConverterNotFound
                                     -3
```

6-188 LIBRARIES

Figure 6-217: <X11/Xutil.h>, Part 3 of 5

```
typedef int XContext;
typedef enum {
       XStringStyle,
       XCompoundTextStyle,
       XTextStyle,
       XStdICCTextStyle
} XICCEncodingStyle;
typedef struct {
       int min_width, min_height;
       int max_width, max_height;
       int width_inc, height_inc;
} XIconSize;
typedef struct {
       char *res_name;
       char *res_class;
} XClassHint;
#define XDestroyImage(ximage)
       ((*((ximage)->f.destroy_image))((ximage)))
#define XGetPixel(ximage, x, y)
       ((*((ximage)->f.get_pixel))((ximage), (x), (y)))
#define XPutPixel(ximage, x, y, pixel)
       ((*((ximage)->f.put_pixel))((ximage), (x), (y), (pixel)))
#define XSubImage(ximage, x, y, width, height)
       ((*(ximage)->f.sub\_image))((ximage), (x), (y), (width), (height)))
#define XAddPixel(ximage, value)
       ((*((ximage)->f.add_pixel))((ximage), (value)))
typedef struct _XComposeStatus {
       XPointer compose_ptr;
       int chars matched;
} XComposeStatus;
```

System Data Interfaces

Figure 6-218: <X11/Xutil.h>, Part 4 of 5

```
#define IsKeypadKey(keysym)
       (((unsigned)(keysym) >= XK_KP_Space) && \
       ((unsigned)(keysym) <= XK_KP_Equal))
#define IsCursorKey(keysym)
       (((unsigned)(keysym) >= XK_Home) && \
       ((unsigned)(keysym) < XK_Select))
#define IsPFKey(keysym)
       (((unsigned)(keysym) >= XK_KP_F1) && \
       ((unsigned)(keysym) <= XK_KP_F4))
#define IsFunctionKey(keysym)
       (((unsigned)(keysym) >= XK_F1) \&\& ((unsigned)(keysym) <= XK_F35))
#define IsMiscFunctionKey(keysym)
       (((unsigned)(keysym) >= XK_Select) && \
       ((unsigned)(keysym) <= XK_Break))
#define IsModifierKey(keysym)
       ((((unsigned)(keysym) >= XK_Shift_L) && \
       ((unsigned)(keysym) <= XK_Hyper_R))
       | | ((unsigned)(keysym) == XK_Mode_switch)
       | ((unsigned)(keysym) == XK_Num_Lock))
typedef void *Region;
#define RectangleOut 0
#define RectangleIn
                             1
#define RectanglePart 2
typedef struct {
       Visual *visual;
       VisualID visualid;
              screen;
       int
       int
              depth;
       int
               class;
       unsigned long red_mask;
       unsigned long green_mask;
       unsigned long blue_mask;
       int
              colormap_size;
       int
               bits_per_rgb;
} XVisualInfo;
```

6-190 LIBRARIES

Figure 6-219: <X11/Xutil.h>, Part 5 of 5

```
#define VisualNoMask
                             0x0
#define VisualIDMask
                             0x1
#define VisualScreenMask
                                     0x2
#define VisualDepthMask
                                     0x4
#define VisualClassMask
                                     0x8
#define VisualRedMaskMask
                                     0x10
#define VisualGreenMaskMask 0x20
#define VisualBlueMaskMask
                                     0x40
#define VisualColormapSizeMask
                                     0x80
#define VisualBitsPerRGBMask 0x100
#define VisualAllMask
                             0x1FF
typedef struct {
       Colormap
                             colormap;
       unsigned long
                             red_max;
       unsigned long
                             red_mult;
       unsigned long
                             green_max;
       unsigned long
                             green_mult;
       unsigned long
                             blue_max;
       unsigned long
                             blue_mult;
       unsigned long
                             base_pixel;
       VisualID
                             visualid;
       XID
                             killid;
} XStandardColormap;
#define ReleaseByFreeingColormap ((XID) 1L)
#define BitmapSuccess
#define BitmapOpenFailed
                                     1
#define BitmapFileInvalid
                                     2
#define BitmapNoMemory
                                     3
#define XCSUCCESS
                                     0
#define XCNOMEM
                                     1
#define XCNOENT
                                     2
#defineXUniqueContext()
                              ((XContext) XrmUniqueQuark())
```

Μ

System Data Interfaces

Motif 1.2 Data Definitions

This section contains standard data definitions that describe system data for the optional Motif 1.2 libraries. These data definitions are referred to by their names in angle brackets: <name.h> and <sys/name.h>. Included in these data definitions are macro definitions and structure definitions. While an ABI-conforming system may provide Motif 1.2 interfaces, it need not contain the actual data definitions referenced here. Programmers should observe that the sources of the structures defined in these data definitions are defined in SVID or the appropriate Motif documentation (see chapter 10 in the Generic ABI).

6-192 LIBRARIES

Figure C 200, 17 (2) 20 1. *			
Figure 6-220: <xm arrowb.h="">*</xm>			
typedef struct _XmArrowButtonClassRec * XmArrowButtonWidgetClass; typedef struct _XmArrowButtonRec * XmArrowButtonWidget;			
The House Control of the Control of the House Control of the Contr			
Figure 6-221: <xm arrowbg.h="">*</xm>			
<pre>typedef struct _XmArrowButtonGadgetClassRec * XmArrowButtonGadgetClass;</pre>			
typedef struct _XmArrowButtonGadgetRec * XmArrowButtonGadget;			
Figure 6-222: <xm bulletinb.h="">*</xm>			
typedef struct _XmBulletinBoardClassRec * XmBulletinBoardWidgetClass;			
typedef struct _XmBulletinBoardRec * XmBulletinBoardWidget;			
Figure 6-223: <xm cascadeb.h="">*</xm>			
typedef struct _XmCascadeButtonRec * XmCascadeButtonWidget;			
typedef struct _XmCascadeButtonRec * XmCascadeButtonWidgetClass;			

Figure 6-224: <Xm/CascadeBG.h>*

typedef struct _XmCascadeButtonGadgetClassRec * XmCascadeButtonGadgetClass; typedef struct _XmCascadeButtonGadgetRec typedef struct _XmCascadeButtonGCacheObjRec

- * XmCascadeButtonGadget;
- * XmCascadeButtonGCacheObject;

Figure 6-225: <Xm/Command.h>*

```
typedef struct _XmCommandClassRec * XmCommandWidgetClass;
```

6-194 **LIBRARIES**

Figure 6-226: <Xm/CutPaste.h>*

```
#define XmClipboardFail
                                0
#define XmClipboardSuccess
                                1
#define XmClipboardTruncate
#define XmClipboardLocked
                                4
#define XmClipboardBadFormat
                                5
#define XmClipboardNoData
                                6
#define ClipboardFail
#define ClipboardSuccess
                                1
#define ClipboardTruncate
                                2
#define ClipboardLocked
                                4
#define ClipboardBadFormat
                                5
#define ClipboardNoData
                                6
typedef struct {
      long DataId;
      long PrivateId;
} XmClipboardPendingRec, *XmClipboardPendingList;
```

Figure 6-227: <Xm/DialogS.h>*

```
Figure 6-228: <Xm/Display.h>*
```

6-196 LIBRARIES

Figure 6-229: <Xm/DragC.h>*, Part 1 of 4

```
#define XmDROP MOVE
                      (1L << 0)
#define XmDROP_COPY
                        (1L << 1)
#define XmDROP_LINK
                        (1L << 2)
#define XmHELP
typedef unsigned int
                        XmID;
#define _XA_MOTIF_DROP "_MOTIF_DROP"
#define _XA_DRAG_FAILURE "_MOTIF_DRAG_FAILURE"
#define _XA_DRAG_SUCCESS "_MOTIF_DRAG_SUCCESS"
enum{
        XmTOP LEVEL ENTER,
                                        XmTOP LEVEL LEAVE,
        XmDRAG_MOTION,
                                        XmDROP_SITE_ENTER,
        XmDROP_SITE_LEAVE,
                                        XmDROP_START,
        XmDROP_FINISH,
                                        XmDRAG_DROP_FINISH,
        XmOPERATION_CHANGED
        } ;
enum{
        XmDROP,
                                        XmDROP_HELP,
        XmDROP_CANCEL,
                                        XmDROP_INTERRUPT
        } ;
```

Figure 6-230: <Xm/DragC.h>*, Part 2 of 4

```
#define XmDROP NOOP
                        0L
enum{
        XmBLEND ALL,
                                        Xmblend State Source,
        XmBLEND_JUST_SOURCE,
                                        XmBLEND_NONE
        } ;
enum{
       XmDROP FAILURE,
                                        XmDROP SUCCESS
        } ;
       XmCR_TOP_LEVEL_ENTER,
                                        XmCR_TOP_LEVEL_LEAVE,
enum{
       XmCR_DRAG_MOTION,
                                        XmCR_DROP_SITE_ENTER,
       XmCR DROP SITE LEAVE,
                                        XmCR DROP START,
                                        XmCR_DRAG_DROP_FINISH,
       XmCR_DROP_FINISH,
       XmCR OPERATION CHANGED,
        _XmNUMBER_DND_CB_REASONS
        } ;
typedef struct _XmDragContextClassRec
                                        *XmDragContextClass;
typedef struct _XmDragContextRec
                                        *XmDragContext;
typedef struct _XmAnyICCCallbackStruct{
    int
                        reason;
   XEvent
                        *event;
    Time
                        timeStamp;
}XmAnyICCCallbackStruct, *XmAnyICCCallback;
typedef struct _XmTopLevelEnterCallbackStruct{
                        reason;
    int
   XEvent
                        *event;
   Time
                        timeStamp;
    Screen
                        *screen;
   Window
                        window;
   Position
                        x, y;
   unsigned char
                        dragProtocolStyle;
   Atom
                        iccHandle;
}XmTopLevelEnterCallbackStruct, *XmTopLevelEnterCallback;
```

6-198 LIBRARIES

Figure 6-231: <Xm/DragC.h>*, Part 3 of 4

```
typedef struct _XmTopLevelLeaveCallbackStruct{
    int
                        reason;
    XEvent
                        *event;
    Time
                        timeStamp;
    Screen
                        *screen;
    Window
                        window;
}XmTopLevelLeaveCallbackStruct, *XmTopLevelLeaveCallback;
typedef struct _XmDropSiteEnterCallbackStruct{
                        reason;
    XEvent
                        *event;
   Time
                        timeStamp;
    unsigned char
                        operation;
    unsigned char
                        operations;
    unsigned char
                        dropSiteStatus;
    Position
                        x, y;
}XmDropSiteEnterCallbackStruct, *XmDropSiteEnterCallback;
typedef struct _XmDropSiteLeaveCallbackStruct{
                        reason;
    XEvent
                        *event;
    Time
                        timeStamp;
}XmDropSiteLeaveCallbackStruct, *XmDropSiteLeaveCallback;
typedef struct _XmDragMotionCallbackStruct{
    int
                        reason;
    XEvent
                        *event;
    Time
                        timeStamp;
    unsigned char
                        operation;
    unsigned char
                        operations;
    unsigned char
                        dropSiteStatus;
    Position
                        x, y;
}XmDragMotionCallbackStruct, *XmDragMotionCallback;
```

Figure 6-232: <Xm/DragC.h>*, Part 4 of 4

```
typedef struct XmOperationChangedCallbackStruct{
    int
                        reason;
    XEvent
                        *event;
    Time
                        timeStamp;
    unsigned char
                        operation;
    unsigned char
                        operations;
    unsigned char
                        dropSiteStatus;
}XmOperationChangedCallbackStruct, *XmOperationChangedCallback;
typedef struct _XmDropStartCallbackStruct{
    int
                        reason;
   XEvent
                        *event;
    Time
                        timeStamp;
    unsigned char
                        operation;
   unsigned char
                        operations;
    unsigned char
                        dropSiteStatus;
    unsigned char
                        dropAction;
    Position
                        x, y;
   Window
                        window;
   Atom
                        iccHandle;
}XmDropStartCallbackStruct, *XmDropStartCallback;
typedef struct _XmDropFinishCallbackStruct{
    int
                        reason;
    XEvent
                        *event;
    Time
                        timeStamp;
   unsigned char
                        operation;
    unsigned char
                        operations;
    unsigned char
                        dropSiteStatus;
   unsigned char
                        dropAction;
    unsigned char
                        completionStatus;
}XmDropFinishCallbackStruct, *XmDropFinishCallback;
typedef struct _XmDragDropFinishCallbackStruct{
    int
                        reason;
   XEvent
                        *event;
    Time
                        timeStamp;
}XmDragDropFinishCallbackStruct, *XmDragDropFinishCallback;
```

6-200 LIBRARIES

```
Figure 6-233: <Xm/DragIcon.h>*
```

Figure 6-234: <Xm/DragOverS.h>*

Figure 6-235: <Xm/DrawingA.h>*

typedef struct _XmDrawingAreaClassRec * XmDrawingAreaWidgetClass; typedef struct _XmDrawingAreaRec * XmDrawingAreaWidget;

Figure 6-236: <Xm/DrawnB.h>*

typedef struct _XmDrawnButtonClassRec *XmDrawnButtonWidgetClass;
typedef struct _XmDrawnButtonRec *XmDrawnButtonWidget;

6-202 LIBRARIES

Figure 6-237: <Xm/DropSMgr.h>*, Part 1 of 2

```
#define XmCR DROP SITE LEAVE MESSAGE
                                       1
#define XmCR_DROP_SITE_ENTER_MESSAGE
                                       2
#define XmCR_DROP_SITE_MOTION_MESSAGE
                                       3
#define XmCR_DROP_MESSAGE
                                       4
#define XmNO_DROP_SITE
                                       1
#define XmINVALID DROP SITE
                                       2
#define XmVALID_DROP_SITE
                                       3
enum { XmDRAG_UNDER_NONE, XmDRAG_UNDER_PIXMAP,
   XmDRAG_UNDER_SHADOW_IN, XmDRAG_UNDER_SHADOW_OUT,
   XmDRAG UNDER HIGHLIGHT };
enum { XmDROP_SITE_SIMPLE, XmDROP_SITE_COMPOSITE,
   XmDROP_SITE_SIMPLE_CLIP_ONLY = 128,
   XmDROP_SITE_COMPOSITE_CLIP_ONLY };
enum { XmABOVE, XmBELOW };
enum { XmDROP_SITE_ACTIVE, XmDROP_SITE_INACTIVE };
typedef struct _XmDragProcCallbackStruct {
      int
                         reason;
      XEvent *
                         event;
      Time
                         timeStamp;
      Widget
                        dragContext;
      Position
                        x, y;
      unsigned char
                        dropSiteStatus;
      unsigned char
                         operation;
      unsigned char
                         operations;
      Boolean
                          animate;
} XmDragProcCallbackStruct, * XmDragProcCallback;
```

Figure 6-238: <Xm/DropSMgr.h>*, Part 2 of 2

```
typedef struct _XmDropProcCallbackStruct {
      int
                          reason;
      XEvent *
                          event;
      Time
                         timeStamp;
      Widget
                          dragContext;
      Position
                          x, y;
      unsigned char
                          dropSiteStatus;
      unsigned char
                          operation;
      unsigned char
                          operations;
      unsigned char
                          dropAction;
} XmDropProcCallbackStruct, * XmDropProcCallback;
typedef struct _XmDropSiteVisualsRec {
      Pixel
                          background;
      Pixel
                          foreground;
      Pixel
                          topShadowColor;
                          topShadowPixmap;
      Pixmap
      Pixel
                          bottomShadowColor;
      Pixmap
                          bottomShadowPixmap;
      Dimension
                          shadowThickness;
      Pixel
                          highlightColor;
      Pixmap
                          highlightPixmap;
      Dimension
                          highlightThickness;
      Dimension
                          borderWidth;
} XmDropSiteVisualsRec, * XmDropSiteVisuals;
typedef struct _XmDropSiteManagerClassRec *XmDropSiteManagerObjectClass;
typedef struct _XmDropSiteManagerRec *XmDropSiteManagerObject;
```

6-204 LIBRARIES

Figure 6-239: <Xm/DropTrans.h>*

```
#define XmTRANSFER_FAILURE 0
#define XmTRANSFER_SUCCESS 1

typedef struct _XmDropTransferClassRec * XmDropTransferObjectClass;
typedef struct _XmDropTransferRec * XmDropTransferObject;

typedef struct _XmDropTransferEntryRec {
    XtPointer client_data;
    Atom target;
} XmDropTransferEntryRec, * XmDropTransferEntry;
```

Figure 6-240: <Xm/FileSB.h>*

```
typedef struct _XmFileSelectionBoxClassRec * XmFileSelectionBoxWidgetClass;
typedef struct _XmFileSelectionBoxRec * XmFileSelectionBoxWidget;
```

Figure 6-241: <Xm/Form.h>*

```
typedef struct _XmFormClassRec * XmFormWidgetClass;
typedef struct _XmFormRec * XmFormWidget;
```

System Data Interfaces

Figure 6-242: <Xm/Frame.h>*

```
typedef struct _XmFrameClassRec * XmFrameWidgetClass;
typedef struct _XmFrameRec * XmFrameWidget;
```

Figure 6-243: <Xm/Label.h>*

Figure 6-244: <Xm/LabelG.h>*

```
typedef struct _XmLabelGadgetClassRec * XmLabelGadgetClass;
typedef struct _XmLabelGadgetRec * XmLabelGadget;
typedef struct _XmLabelGCacheObjRec * XmLabelGCacheObject;
```

6-206 LIBRARIES

```
Figure 6-245: <Xm/List.h>*
```

```
#define XmINITIAL 0
#define XmADDITION 1
#define XmMODIFICATION 2

typedef struct _XmListClassRec * XmListWidgetClass;
typedef struct _XmListRec * XmListWidget;
```

Figure 6-246: <Xm/MainW.h>*

```
typedef struct _XmMainWindowClassRec * XmMainWindowWidgetClass;
typedef struct _XmMainWindowRec * XmMainWindowWidget;
```

Figure 6-247: <Xm/MenuShell.h>*

System Data Interfaces

Figure 6-248: <Xm/MessageB.h>*

```
typedef struct _XmMessageBoxClassRec * XmMessageBoxWidgetClass;
typedef struct _XmMessageBoxRec * XmMessageBoxWidget;
```

6-208 LIBRARIES

Figure 6-249: <Mrm/MrmPublic.h>*, Part 1 of 3

#define	MrmSUCCESS	1
#define	MrmCREATE_NEW	3
#define	MrmINDEX_RETRY	5
#define	MrmINDEX_GT	7
#define	MrmINDEX_LT	9
#define	MrmPARTIAL_SUCCESS	11
#define	MrmFAILURE	0
#define	MrmNOT_FOUND	2
#define	MrmEXISTS	4
#define	MrmNUL_GROUP	6
#define	MrmNUL_TYPE	8
#define	MrmWRONG_GROUP	10
#define	MrmWRONG_TYPE	12
#define	MrmOUT_OF_RANGE	14
#define	MrmBAD_RECORD	16
	MrmNULL_DATA	18
#define	MrmBAD_DATA_INDEX	20
#define	MrmBAD_ORDER	22
#define	MrmBAD_CONTEXT	24
#define	MrmNOT_VALID	26
#define	MrmBAD_BTREE	28
#define	MrmBAD_WIDGET_REC	30
#define	MrmBAD_CLASS_TYPE	32
#define	MrmNO_CLASS_NAME	34
#define	MrmTOO_MANY	36
#define	MrmBAD_IF_MODULE	38
#define	MrmNULL_DESC	40
#define	MrmOUT_OF_BOUNDS	42
#define	MrmBAD_COMPRESS	44
#define	MrmBAD_ARG_TYPE	46
#define	MrmNOT_IMP	48
#define	MrmNULL_INDEX	50
#define	MrmBAD_KEY_TYPE	52
#define	MrmBAD_CALLBACK	54

Figure 6-250: <Mrm/MrmPublic.h>*, Part 2 of 3

#define	MrmNULL_ROUTINE	56
#define	MrmVEC_TOO_BIG	58
#define	MrmBAD_HIERARCHY	60
#define	MrmBAD_CLASS_CODE	62
#define	MrmDISPLAY_NOT_OPENED	63
#define	MrmEOF	64
#define	MrmUNRESOLVED_REFS	65
#define	MrmNcreateCallback	"createCallback"
#define	MrmCR_CREATE	XmCR_CREATE
#define	MrmwcUnknown	1
#define	MrmRtypeMin	1
#define	MrmRtypeInteger	1
#define	MrmRtypeBoolean	2
#define	MrmRtypeChar8	3
#define	MrmRtypeChar8Vector	4
#define	MrmRtypeCString	5
#define	MrmRtypeCStringVector	6
#define	MrmRtypeFloat	7
#define	MrmRtypeCallback	9
#define	MrmRtypePixmapImage	10
#define	MrmRtypePixmapDDIF	11
#define	MrmRtypeResource	12
#define	MrmRtypeNull	13
#define	MrmRtypeAddrName	14
#define	MrmRtypeIconImage	15
#define	MrmRtypeFont	16
#define	MrmRtypeFontList	17
#define	MrmRtypeColor	18
#define	MrmRtypeColorTable	19
#define	MrmRtypeAny	20
#define	MrmRtypeTransTable	21
#define	MrmRtypeClassRecName	22
#define	MrmRtypeIntegerVector	23

6-210 LIBRARIES

Figure 6-251: <Mrm/MrmPublic.h>*, Part 3 of 3

```
#define MrmRtypeXBitmapFile
                                24
#define MrmRtypeCountedVector
                                25
#define MrmRtypeKeysym
                                26
                                27
#define MrmRtypeSingleFloat
#define MrmRtypeWideCharacter
                                28
#define MrmRtypeFontSet
                                29
#define MrmRtypeMax
                                30
typedef short int
                             MrmCode ;
typedef unsigned char
                             MrmSCode ;
typedef unsigned short int
                             MrmOffset ;
typedef short int
                             MrmType ;
typedef unsigned short int
                             MrmSize ;
typedef short int
                             MrmCount ;
typedef unsigned char
                             MrmFlag ;
typedef long int
                             MrmResource_id ;
typedef short int
                             MrmGroup ;
#define MrmMaxResourceSize
                                65535
#define MrmOsOpenParamVersion
                                 1
typedef struct {
        Cardinal
                                version;
        char
                                *default fname;
        union {
            unsigned long
                                related_nam;
            Boolean
                                clobber_flg;
        } nam_flg;
        Display
                                *display;
} MrmOsOpenParam, *MrmOsOpenParamPtr ;
typedef struct MrmHierarchyDescStruct *MrmHierarchy;
typedef struct {
        String
                        name ;
        XtPointer
                        value ;
} MRMRegisterArg, MrmRegisterArg, *MrmRegisterArglist ;
#define URMwcUnknown
```

Figure 6-252: <Xm/MwmUtil.h>*, Part 1 of 3

```
typedef struct
   long
                flags;
   long
                functions;
   long
                decorations;
    int
                input mode;
   long
                status;
} MotifWmHints;
typedef MotifWmHints
                        MwmHints;
#define MWM HINTS FUNCTIONS
                                 (1L << 0)
#define MWM_HINTS_DECORATIONS
                                 (1L << 1)
#define MWM_HINTS_INPUT_MODE
                                 (1L << 2)
#define MWM_HINTS_STATUS
                                 (1L << 3)
                                 (1L << 0)
#define MWM FUNC ALL
#define MWM FUNC RESIZE
                                 (1L << 1)
#define MWM_FUNC_MOVE
                                 (1L << 2)
#define MWM_FUNC_MINIMIZE
                                 (1L << 3)
#define MWM_FUNC_MAXIMIZE
                                 (1L << 4)
#define MWM_FUNC_CLOSE
                                 (1L << 5)
#define MWM_DECOR_ALL
                                 (1L << 0)
#define MWM DECOR BORDER
                                 (1L << 1)
#define MWM_DECOR_RESIZEH
                                 (1L << 2)
#define MWM_DECOR_TITLE
                                 (1L << 3)
#define MWM DECOR MENU
                                 (1L << 4)
#define MWM DECOR MINIMIZE
                                 (1L << 5)
#define MWM_DECOR_MAXIMIZE
                                 (1L << 6)
#define MWM_INPUT_MODELESS
                                                 0
#define MWM_INPUT_PRIMARY_APPLICATION_MODAL
                                                 1
#define MWM INPUT SYSTEM MODAL
                                                 2
#define MWM_INPUT_FULL_APPLICATION_MODAL
                                                 3
```

6-212 LIBRARIES

Figure 6-253: <Xm/MwmUtil.h>*, Part 2 of 3

```
#define MWM TEAROFF WINDOW
                                (1L << 0)
#define MWM_INPUT_APPLICATION_MODAL
                                        MWM_INPUT_PRIMARY_APPLICATION_MODAL
typedef struct
                flags;
    long
               wm_window;
   Window
} MotifWmInfo;
typedef MotifWmInfo
                       MwmInfo;
#define MWM INFO STARTUP STANDARD
                                      (1L << 0)
#define MWM_INFO_STARTUP_CUSTOM
                                      (1L << 1)
typedef struct
   CARD32
               flags;
              functions;
   CARD32
   CARD32
              decorations;
   INT32
               inputMode;
   CARD32
               status;
} PropMotifWmHints;
typedef PropMotifWmHints
                               PropMwmHints;
#define PROP_MOTIF_WM_HINTS_ELEMENTS
#define PROP_MWM_HINTS_ELEMENTS
                                        PROP_MOTIF_WM_HINTS_ELEMENTS
#define _XA_MOTIF_WM_HINTS
                                "_MOTIF_WM_HINTS"
#define _XA_MWM_HINTS
                                _XA_MOTIF_WM_HINTS
#define _XA_MOTIF_WM_MESSAGES "_MOTIF_WM_MESSAGES"
#define _XA_MWM_MESSAGES
                               _XA_MOTIF_WM_MESSAGES
#define _XA_MOTIF_WM_OFFSET
                                "_MOTIF_WM_OFFSET"
```

Figure 6-254: <Xm/MwmUtil.h>*, Part 3 of 3

```
#define XA MOTIF WM MENU
                                " MOTIF WM MENU"
#define _XA_MWM_MENU
                                _XA_MOTIF_WM_MENU
typedef struct
   CARD32 flags;
   CARD32 wmWindow;
} PropMotifWmInfo;
typedef PropMotifWmInfo PropMwmInfo;
#define PROP_MOTIF_WM_INFO_ELEMENTS
#define PROP_MWM_INFO_ELEMENTS
                                        PROP_MOTIF_WM_INFO_ELEMENTS
#define _XA_MOTIF_WM_INFO
                                "_MOTIF_WM_INFO"
#define _XA_MWM_INFO
                                _XA_MOTIF_WM_INFO
#define _XA_MOTIF_BINDINGS
                                "_MOTIF_BINDINGS"
```

Figure 6-255: <Xm/PanedW.h>*

6-214 LIBRARIES

```
Figure 6-256: <Xm/PushB.h>*
```

```
typedef struct _XmPushButtonClassRec *XmPushButtonWidgetClass;
typedef struct _XmPushButtonRec *XmPushButtonWidget;
```

Figure 6-257: <Xm/PushBG.h>*

Figure 6-258: <Xm/RepType.h>*

System Data Interfaces

```
Figure 6-259: <Xm/RowColumn.h>*
typedef struct _XmRowColumnClassRec * XmRowColumnWidgetClass;
typedef struct _XmRowColumnRec
                                  * XmRowColumnWidget;
Figure 6-260: <Xm/Scale.h>*
    typedef struct _XmScaleClassRec * XmScaleWidgetClass;
    Figure 6-261: <Xm/Screen.h>*
      typedef struct _XmScreenRec
                                    *XmScreen;
      typedef struct _XmScreenClassRec *XmScreenClass;
Figure 6-262: <Xm/ScrollBar.h>*
typedef struct _XmScrollBarClassRec * XmScrollBarWidgetClass;
typedef struct _XmScrollBarRec
                                  * XmScrollBarWidget;
```

6-216 LIBRARIES

Figure 6-263: <xm scrolledw.h="">*</xm>
<pre>typedef struct _XmScrolledWindowClassRec * XmScrolledWindowWidgetClass; typedef struct _XmScrolledWindowRec * XmScrolledWindowWidget;</pre>
Figure 6-264: <xm selectiob.h="">*</xm>
<pre>typedef struct _XmSelectionBoxClassRec * XmSelectionBoxWidgetClass; typedef struct _XmSelectionBoxRec * XmSelectionBoxWidget;</pre>
Figure 6-265: <xm separatog.h="">*</xm>
<pre>typedef struct _XmSeparatorGadgetClassRec * XmSeparatorGadgetClass; typedef struct _XmSeparatorGadgetRec * XmSeparatorGadget; typedef struct _XmSeparatorGCacheObjRec * XmSeparatorGCacheObject;</pre>
Figure 6-266: <xm separator.h="">*</xm>
<pre>typedef struct _XmSeparatorClassRec * XmSeparatorWidgetClass; typedef struct _XmSeparatorRec * XmSeparatorWidget;</pre>

```
Figure 6-267: <Xm/Text.h>*
     typedef struct _XmTextSourceRec *XmTextSource;
     typedef struct _XmTextClassRec *XmTextWidgetClass;
     typedef struct _XmTextRec
                                      *XmTextWidget;
Figure 6-268: <Xm/TextF.h>*
 typedef struct _XmTextFieldClassRec *XmTextFieldWidgetClass;
 typedef struct _XmTextFieldRec
                                      *XmTextFieldWidget;
Figure 6-269: <Xm/ToggleB.h>*
typedef struct _XmToggleButtonClassRec *XmToggleButtonWidgetClass;
typedef struct _XmToggleButtonRec
                                       *XmToggleButtonWidget;
Figure 6-270: <Xm/ToggleBG.h>*
typedef struct _XmToggleButtonGadgetClassRec
                                                  *XmToggleButtonGadgetClass;
typedef struct _XmToggleButtonGadgetRec
                                                  *XmToggleButtonGadget;
typedef struct _XmToggleButtonGCacheObjRec
                                                  *XmToggleButtonGCacheObject;
```

6-218 LIBRARIES

Figure 6-271: <Xm/VendorS.h>*

Figure 6-272: <Xm/VirtKeys.h>*, Part 1 of 2

#define	_OSF_Keysyms	
#define #define #define #define	osfXK_BackSpace osfXK_Insert osfXK_Delete osfXK_Copy osfXK_Cut osfXK_Paste	0x1004FF08 0x1004FF63 0x1004FFFF 0x1004FF02 0x1004FF03 0x1004FF04
#define	osfXK_AddMode osfXK_PrimaryPaste osfXK_QuickPaste	0x1004FF31 0x1004FF32 0x1004FF33
#define #define	osfXK_PageLeft osfXK_PageUp osfXK_PageDown osfXK_PageRight	0x1004FF40 0x1004FF41 0x1004FF42 0x1004FF43

Figure 6-273: <Xm/VirtKeys.h>*, Part 2 of 2

#define	osfXK_EndLine	0x1004FF57
#define	osfXK_BeginLine	0x1004FF58
#define	osfXK_Activate	$0 \times 1004 \text{FF}44$
#define	osfXK_MenuBar	0x1004FF45
#define	osfXK_Clear	0x1004FF0B
#define	osfXK_Cancel	0x1004FF69
#define	osfXK_Help	0x1004FF6A
#define	osfXK_Menu	0x1004FF67
#define	osfXK_Select	0x1004FF60
#define	osfXK_Undo	0x1004FF65
#define	osfXK_Left	0x1004FF51
#define	osfXK_Up	0x1004FF52
#define	osfXK_Right	0x1004FF53
#define	osfXK_Down	0x1004FF54

6-220 LIBRARIES

Figure 6-274: <Xm/Xm.h>*, Part 1 of 14

```
#define XmUNSPECIFIED PIXMAP
#define XmSTRING OS CHARSET
                                       XmSTRING ISO8859 1
#define XmFALLBACK_CHARSET
                                       XmSTRING_ISO8859_1
#define XmDEFAULT FONT
                                       XmSDEFAULT FONT
#define XmDEFAULT BACKGROUND
                                       _XmSDEFAULT_BACKGROUND
                                       20
#define XmDEFAULT DARK THRESHOLD
#define XmDEFAULT_LIGHT_THRESHOLD
                                       90
#define XmDEFAULT_FOREGROUND_THRESHOLD 70
typedef enum{ XmFONT IS FONT, XmFONT IS FONTSET } XmFontType;
enum{
       XmSTRING DIRECTION L TO R,
                                       XmSTRING DIRECTION R TO L
        } ;
#define XmSTRING_DIRECTION_DEFAULT ((XmStringDirection) 255)
typedef unsigned char * XmString;
typedef XmString *
                       XmStringTable;
typedef char *
                       XmStringCharSet;
typedef unsigned char XmStringComponentType;
typedef unsigned char XmStringDirection;
typedef struct _XmFontListRec
                                   *XmFontListEntry;
typedef struct _XmFontListRec
                                   *XmFontList;
typedef struct __XmStringContextRec *_XmStringContext;
typedef struct __XmStringRec
                                  * XmString;
typedef struct _XmtStringContextRec *XmStringContext;
typedef struct _XmFontListContextRec *XmFontContext;
       XmSTRING_COMPONENT_UNKNOWN,
                                       XmSTRING_COMPONENT_CHARSET,
enum{
                                       XmSTRING_COMPONENT_DIRECTION,
       XmSTRING_COMPONENT_TEXT,
       XmSTRING_COMPONENT_SEPARATOR, XmSTRING_COMPONENT_LOCALE_TEXT
        } ;
```

Figure 6-275: <Xm/Xm.h>*, Part 2 of 14

```
#define XmSTRING COMPONENT END
                                    ((XmStringComponentType) 126)
#define XmSTRING_COMPONENT_USER_BEGIN
                                    ((XmStringComponentType) 128)
#define XmSTRING_COMPONENT_USER_END
                                    ((XmStringComponentType) 255)
typedef struct _XmPrimitiveClassRec * XmPrimitiveWidgetClass;
typedef struct _XmGadgetClassRec * XmGadgetClass;
typedef struct _XmGadgetRec
                             * XmGadget;
typedef struct _XmManagerClassRec * XmManagerWidgetClass;
enum{
       XmCHANGE ALL,
                                   XmCHANGE NONE,
       XmCHANGE_WIDTH,
                                   XmCHANGE_HEIGHT
       } ;
enum{
       XmPIXELS,
                                   Xm100TH MILLIMETERS,
       Xm1000TH_INCHES,
                                   Xm100TH_POINTS,
       Xm100TH_FONT_UNITS
       } ;
enum{
       XmDESTROY,
                                   XmUNMAP,
       XmDO_NOTHING
       } ;
enum{
       XmEXPLICIT,
                                   XmPOINTER
       } ;
enum{
       XmNONE,
                                    XmTAB_GROUP,
       XmSTICKY_TAB_GROUP,
                                   XmEXCLUSIVE_TAB_GROUP
       } ;
#define XmDYNAMIC DEFAULT TAB GROUP
                                    ((XmNavigationType) 255)
```

6-222 LIBRARIES

Figure 6-276: <Xm/Xm.h>*, Part 3 of 14

```
enum{
                            XmBELL = 1
        } ;
        XmNO_ORIENTATION,
                                         XmVERTICAL,
enum{
        XmHORIZONTAL
        } ;
enum{
        XmWORK_AREA,
                                         XmMENU_BAR,
        XmMENU_PULLDOWN,
                                         XmMENU_POPUP,
        XmMENU_OPTION
        } ;
enum{
        XmNO_PACKING,
                                         XmPACK_TIGHT,
        XmPACK_COLUMN,
                                         XmPACK_NONE
        } ;
        XmALIGNMENT_CONTENTS_TOP = 3,
enum{
        XmALIGNMENT_CONTENTS_BOTTOM
        } ;
        XmTEAR_OFF_ENABLED,
enum{
                                         XmTEAR_OFF_DISABLED
        } ;
enum{
        XmUNPOST,
                                         XmUNPOST_AND_REPLAY
        } ;
enum{
        XmLAST_POSITION = -1,
                                         XmFIRST_POSITION
        } ;
        XmALIGNMENT_BEGINNING,
                                        XmALIGNMENT_CENTER,
enum{
        XmALIGNMENT_END
        } ;
```

Figure 6-277: <Xm/Xm.h>*, Part 4 of 14

```
XmALIGNMENT_BASELINE_TOP,
enum{
        Xmalignment_baseline_bottom = 2, Xmalignment_widget_top,
        XmALIGNMENT_WIDGET_BOTTOM
        } ;
        XmFRAME_GENERIC_CHILD,
enum{
                                         XmFRAME_WORKAREA_CHILD,
        XmFRAME_TITLE_CHILD
        } ;
        XmN_OF_MANY = 1,
enum{
                                         XmONE_OF_MANY
        } ;
enum{
        XmATTACH_NONE,
                                         XmATTACH_FORM,
        XmATTACH_OPPOSITE_FORM,
                                         XmATTACH_WIDGET,
        XmATTACH_OPPOSITE_WIDGET,
                                         XmATTACH_POSITION,
        XmATTACH_SELF
        } ;
enum{
        XmRESIZE_NONE,
                                         XmRESIZE_GROW,
        XmRESIZE_ANY
        } ;
```

6-224 LIBRARIES

Figure 6-278: <Xm/Xm.h>*, Part 5 of 15

```
enum{
        XmCR NONE,
                                         XmCR HELP,
        XmCR_VALUE_CHANGED,
                                         XmCR_INCREMENT,
        XmCR DECREMENT,
                                         XmCR PAGE INCREMENT,
        XmCR_PAGE_DECREMENT,
                                         XmCR_TO_TOP,
        XmCR_TO_BOTTOM,
                                         XmCR DRAG,
        XmCR_ACTIVATE,
                                         XmCR ARM,
        XmCR DISARM,
                                         XmCR MAP = 16,
        XmCR_UNMAP,
                                         XmCR_FOCUS,
        XmCR_LOSING_FOCUS,
                                         XmCR_MODIFYING_TEXT_VALUE,
        XmCR_MOVING_INSERT_CURSOR,
                                         XmCR_EXECUTE,
        XmCR_SINGLE_SELECT,
                                         XmCR_MULTIPLE_SELECT,
        XmCR EXTENDED SELECT,
                                         XmCR BROWSE SELECT,
        XmCR_DEFAULT_ACTION,
                                         XmCR_CLIPBOARD_DATA_REQUEST,
        XmCR CLIPBOARD DATA DELETE,
                                         XmCR CASCADING,
        XmCR_OK,
                                         XmCR_CANCEL,
                                         XmCR_NO_MATCH,
        XmCR\_APPLY = 34,
        XmCR_COMMAND_ENTERED,
                                         XmCR_COMMAND_CHANGED,
        XmCR EXPOSE,
                                         XmCR RESIZE,
        XmCR_INPUT,
                                         XmCR_GAIN_PRIMARY,
        XmCR_LOSE_PRIMARY,
                                         XmCR_CREATE,
        XmCR_TEAR_OFF_ACTIVATE,
                                         XmCR_TEAR_OFF_DEACTIVATE,
        XmCR_OBSCURED_TRAVERSAL
        } ;
typedef struct
    int
            reason;
   XEvent *event;
} XmAnyCallbackStruct;
typedef struct
    int
            reason;
   XEvent *event;
            click_count;
    int
} XmArrowButtonCallbackStruct;
```

Figure 6-279: <Xm/Xm.h>*, Part 6 of 14

```
typedef struct
    int
            reason;
    XEvent *event;
    Window window;
} XmDrawingAreaCallbackStruct;
typedef struct
    int
            reason;
    XEvent *event;
    Window window;
    int
            click_count;
} XmDrawnButtonCallbackStruct;
typedef struct
    int
           reason;
    XEvent *event;
    int
           click_count;
} XmPushButtonCallbackStruct;
typedef struct
    int
            reason;
    XEvent *event;
    Widget widget;
            *data;
    char
    char
            *callbackstruct;
} XmRowColumnCallbackStruct;
```

6-226 LIBRARIES

Figure 6-280: <Xm/Xm.h>*, Part 7 of 14

```
typedef struct
   int reason;
  XEvent * event;
   int value;
   int pixel;
} XmScrollBarCallbackStruct;
typedef struct
   int reason;
  XEvent * event;
   int set;
} XmToggleButtonCallbackStruct;
typedef struct
             reason;
   int
  XEvent
            *event;
  XmString item;
             item_length;
   int
   int
            item_position;
  XmString *selected_items;
   int
             selected_item_count;
   int
             *selected_item_positions;
   char
             selection_type;
} XmListCallbackStruct;
typedef struct
    int reason;
   XEvent
                *event;
   XmString
                value;
                length;
} XmSelectionBoxCallbackStruct;
```

6-227

Figure 6-281: <Xm/Xm.h>*, Part 8 of 14

```
typedef struct
   int reason;
   XEvent
           *event;
   XmString value;
               length;
   int
} XmCommandCallbackStruct;
typedef struct
              reason;
   int
   XEvent
              *event;
   XmString value;
   int
               length;
   XmString mask;
   int
              mask_length;
              dir ;
   XmString
   int
              dir_length ;
   XmString pattern;
              pattern_length ;
} XmFileSelectionBoxCallbackStruct;
typedef struct
   int reason;
  XEvent * event;
  int value;
} XmScaleCallbackStruct;
       XmMULTICLICK_DISCARD,
enum{
                                     XmMULTICLICK_KEEP
       } ;
       XmSHADOW_IN = 7,
                                      XmSHADOW_OUT
enum{
       } ;
```

6-228 LIBRARIES

Figure 6-282: <Xm/Xm.h>*, Part 9 of 14

```
XmARROW UP,
enum{
                                         XmARROW DOWN,
        XmARROW_LEFT,
                                         XmARROW_RIGHT
        } ;
enum{
        XmNO_LINE,
                                         XmSINGLE_LINE,
        XmDOUBLE_LINE,
                                         XmSINGLE_DASHED_LINE,
        XmDOUBLE_DASHED_LINE,
                                         XmSHADOW_ETCHED_IN,
        XmSHADOW_ETCHED_OUT,
                                         XmSHADOW_ETCHED_IN_DASH,
        XmSHADOW_ETCHED_OUT_DASH,
                                         XmINVALID_SEPARATOR_TYPE
        } ;
enum{
        XmPIXMAP = 1,
                                         XmSTRING
        } ;
enum{
        XmWINDOW,
        XmCURSOR = 2
        } ;
enum{
        XmMAX_ON_TOP,
                                         XmMAX_ON_BOTTOM,
        XmMAX_ON_LEFT,
                                         XmMAX_ON_RIGHT
        } ;
        XmSINGLE SELECT,
                                         XmMULTIPLE SELECT,
enum{
        XmEXTENDED_SELECT,
                                         XmBROWSE_SELECT
        } ;
enum{
        XmSTATIC,
                                         XmDYNAMIC
        } ;
enum{
        XmVARIABLE,
                                         XmCONSTANT,
        XmRESIZE_IF_POSSIBLE
        } ;
enum{
        XmAUTOMATIC,
                                         XmAPPLICATION DEFINED
        } ;
```

Figure 6-283: <Xm/Xm.h>*, Part 10 of 14

```
enum{ XmAS NEEDED = 1
        } ;
#define SW_TOP
                        1
#define SW_BOTTOM
#define SW_LEFT
                        2
#define SW_RIGHT
#define XmTOP_LEFT
                     (SW_TOP | SW_LEFT)
#define XmBOTTOM_LEFT
                       (SW_BOTTOM | SW_LEFT)
#define XmTOP_RIGHT (SW_TOP | SW_RIGHT)
#define XmBOTTOM_RIGHT (SW_BOTTOM | SW_RIGHT)
enum{
        XmCOMMAND_ABOVE_WORKSPACE,
                                        XmCOMMAND_BELOW_WORKSPACE
        } ;
       XmMULTI_LINE_EDIT,
enum{
                                        XmSINGLE_LINE_EDIT
        } ;
{\tt typedef\ enum} \{
        XmTEXT_FORWARD,
       XmTEXT_BACKWARD
        } XmTextDirection;
typedef long XmTextPosition;
typedef Atom XmTextFormat;
#define XmFMT_8_BIT
                        ((XmTextFormat) XA_STRING)
#define XmFMT_16_BIT
                        ((XmTextFormat) 2)
#define FMT8BIT
                        XmFMT_8_BIT
#define FMT16BIT
                        XmFMT_16_BIT
```

6-230 LIBRARIES

Figure 6-284: <Xm/Xm.h>*, Part 11 of 14

```
typedef enum{
        XmSELECT_POSITION,
                                         XmSELECT_WHITESPACE,
        XmSELECT_WORD,
                                         XmSELECT LINE,
        XmSELECT_ALL,
                                         XmSELECT_PARAGRAPH
        } XmTextScanType ;
typedef enum{
        XmHIGHLIGHT_NORMAL,
                                         XmHIGHLIGHT_SELECTED,
        {\tt XmHIGHLIGHT\_SECONDARY\_SELECTED}
        } XmHighlightMode;
typedef struct {
   char *ptr;
    int length;
    XmTextFormat format;
} XmTextBlockRec, *XmTextBlock;
typedef struct
    int reason;
   XEvent *event;
   Boolean doit;
    long currInsert, newInsert;
    long startPos, endPos;
    XmTextBlock text;
} XmTextVerifyCallbackStruct, *XmTextVerifyPtr;
typedef struct {
   wchar_t *wcsptr;
    int length;
} XmTextBlockRecWcs, *XmTextBlockWcs;
```

System Data Interfaces

Figure 6-285: <Xm/Xm.h>*, Part 12 of 14

```
typedef struct
    int reason;
    XEvent *event;
    Boolean doit;
    long currInsert, newInsert;
    long startPos, endPos;
    XmTextBlockWcs text;
} XmTextVerifyCallbackStructWcs, *XmTextVerifyPtrWcs;
#define XmTextGetTopPosition
                                             XmTextGetTopCharacter
#define XmTextSetTopPosition
                                             XmTextSetTopCharacter
#define XmCOPY FAILED
                                0
#define XmCOPY_SUCCEEDED
                                1
#define XmCOPY_TRUNCATED
        XmDIALOG NONE,
                                         XmDIALOG APPLY BUTTON,
enum{
        XmDIALOG_CANCEL_BUTTON,
                                         XmDIALOG_DEFAULT_BUTTON,
        XmDIALOG_OK_BUTTON,
                                         XmDIALOG_FILTER_LABEL,
        XmDIALOG_FILTER_TEXT,
                                         XmDIALOG_HELP_BUTTON,
        XmDIALOG_LIST,
                                         XmDIALOG_LIST_LABEL,
        XmDIALOG MESSAGE LABEL,
                                        XmDIALOG SELECTION LABEL,
        XmDIALOG_SYMBOL_LABEL,
                                        XmDIALOG_TEXT,
        XmDIALOG SEPARATOR,
                                         XmDIALOG_DIR_LIST,
        XmDIALOG_DIR_LIST_LABEL
        } ;
#define XmDIALOG HISTORY LIST
                                         XmDIALOG LIST
#define XmDIALOG_PROMPT_LABEL
                                         XmDIALOG_SELECTION_LABEL
#define XmDIALOG_VALUE_TEXT
                                         XmDIALOG TEXT
#define XmDIALOG_COMMAND_TEXT
                                        XmDIALOG_TEXT
#define XmDIALOG_FILE_LIST
                                        XmDIALOG_LIST
#define XmDIALOG FILE LIST LABEL
                                        XmDIALOG LIST LABEL
```

6-232 LIBRARIES

Figure 6-286: <Xm/Xm.h>*, Part 13 of 14

```
enum{
        XmDIALOG MODELESS,
                                        XmDIALOG PRIMARY APPLICATION MODAL,
        XmDIALOG_FULL_APPLICATION_MODAL,XmDIALOG_SYSTEM_MODAL
#define XmDIALOG_APPLICATION_MODAL
                                        XmDIALOG_PRIMARY_APPLICATION_MODAL
        XmPLACE TOP,
enum{
                                        XmPLACE ABOVE SELECTION,
        XmPLACE_BELOW_SELECTION
        } ;
#define XmFILE_DIRECTORY (1 << 0)</pre>
#define XmFILE REGULAR
                        (1 << 1)
#define XmFILE_ANY_TYPE (XmFILE_DIRECTORY | XmFILE_REGULAR)
        XmDIALOG_WORK_AREA,
                                        XmDIALOG_PROMPT,
enum{
        XmDIALOG SELECTION,
                                        XmDIALOG COMMAND,
        XmDIALOG_FILE_SELECTION
        } ;
enum{
        XmDIALOG_TEMPLATE,
                                        XmDIALOG ERROR,
        XmDIALOG_INFORMATION,
                                        XmDIALOG_MESSAGE,
        XmDIALOG_QUESTION,
                                        XmDIALOG_WARNING,
        XmDIALOG WORKING
        } ;
typedef enum{
        XmVISIBILITY_UNOBSCURED,
                                        XmVISIBILITY_PARTIALLY_OBSCURED,
        XmVISIBILITY_FULLY_OBSCURED
        } XmVisibility ;
typedef enum{
        XmTRAVERSE_CURRENT,
                                         XmTRAVERSE_NEXT,
        XmTRAVERSE_PREV,
                                        XmTRAVERSE_HOME,
        XmTRAVERSE NEXT TAB GROUP,
                                         XmTRAVERSE PREV TAB GROUP,
        XmTRAVERSE_UP,
                                        XmTRAVERSE_DOWN,
        XmTRAVERSE LEFT,
                                         XmTRAVERSE RIGHT
        } XmTraversalDirection ;
```

System Data Interfaces

Figure 6-287: <Xm/Xm.h>*, Part 14 of 14

```
typedef struct _XmTraverseObscuredCallbackStruct
        int
                               reason;
       XEvent *
                               event ;
       Widget
                               traversal_destination ;
        XmTraversalDirection
                               direction ;
        } XmTraverseObscuredCallbackStruct ;
typedef unsigned char
                       XmNavigationType;
typedef unsigned char XmButtonType;
typedef XmButtonType * XmButtonTypeTable;
typedef KeySym * XmKeySymTable;
typedef XmStringCharSet * XmStringCharSetTable;
       XmPUSHBUTTON = 1,
                                       XmTOGGLEBUTTON,
enum{
       XmRADIOBUTTON,
                                       XmCASCADEBUTTON,
       XmSEPARATOR,
                                       XmDOUBLE_SEPARATOR,
       XmTITLE
        } ;
#define XmCHECKBUTTON
                                        XmTOGGLEBUTTON
typedef struct _XmSecondaryResourceDataRec{
    XmResourceBaseProc base_proc;
    XtPointer
                       client_data;
    String
                      name;
    String
                      res class;
   XtResourceList
                      resources;
    Cardinal
                       num_resources;
}XmSecondaryResourceDataRec, *XmSecondaryResourceData;
typedef long XmOffset;
typedef XmOffset *XmOffsetPtr;
```

6-234 LIBRARIES

Figure 6-288: <Xm/XmStrDefs.h>*, Part 1 of 34

```
#define XmS ""
#define XmCAccelerator "Accelerator"
#define XmCAcceleratorText "AcceleratorText"
#define XmCAdjustLast "AdjustLast"
#define XmCAdjustMargin "AdjustMargin"
#define XmCAlignment "Alignment"
#define XmCAllowOverlap "AllowOverlap"
#define XmCAnimationMask "AnimationMask"
#define XmCAnimationPixmap "AnimationPixmap"
#define XmCAnimationPixmapDepth "AnimationPixmapDepth"
#define XmCAnimationStyle "AnimationStyle"
#define XmCApplyLabelString "ApplyLabelString"
#define XmCArmCallback "ArmCallback"
#define XmCArmColor "ArmColor"
#define XmCArmPixmap "ArmPixmap"
#define XmCArrowDirection "ArrowDirection"
#define XmCAttachment "Attachment"
#define XmCAudibleWarning "AudibleWarning"
#define XmCAutoShowCursorPosition "AutoShowCursorPosition"
#define XmCAutoUnmanage "AutoUnmanage"
#define XmCAutomaticSelection "AutomaticSelection"
#define XmCAvailability "Availability"
#define XmCBackgroundPixmap "BackgroundPixmap"
#define XmCBlendModel "BlendModel"
#define XmCBlinkRate "BlinkRate"
#define XmCBottomShadowColor "BottomShadowColor"
#define XmCBottomShadowPixmap "BottomShadowPixmap"
#define XmCButtonAcceleratorText "ButtonAcceleratorText"
#define XmCButtonAccelerators "ButtonAccelerators"
#define XmCButtonCount "ButtonCount"
#define XmCButtonFontList "ButtonFontList"
#define XmCButtonMnemonicCharSets "ButtonMnemonicCharSets"
#define XmCButtonMnemonics "ButtonMnemonics"
#define XmCButtonSet "ButtonSet"
#define XmCButtonType "ButtonType"
```

Figure 6-289: <Xm/XmStrDefs.h>*, Part 2 of 34

```
#define XmCButtons "Buttons"
#define XmCCancelLabelString "CancelLabelString"
#define XmCChildHorizontalAlignment "ChildHorizontalAlignment"
#define XmCChildHorizontalSpacing "ChildHorizontalSpacing"
#define XmCChildPlacement "ChildPlacement"
#define XmCChildType "ChildType"
#define XmCChildVerticalAlignment "ChildVerticalAlignment"
#define XmCChildren "Children"
#define XmCClientData "ClientData"
#define XmCClipWindow "ClipWindow"
#define XmCColumns "Columns"
#define XmCCommandWindow "CommandWindow"
#define XmCCommandWindowLocation "CommandWindowLocation"
#define XmCConvertProc "ConvertProc"
#define XmCCursorBackground "CursorBackground"
#define XmCCursorForeground "CursorForeground"
#define XmCCursorPosition "CursorPosition"
#define XmCCursorPositionVisible "CursorPositionVisible"
#define XmCDarkThreshold "DarkThreshold"
#define XmCDecimalPoints "DecimalPoints"
#define XmCDefaultButtonShadowThickness "DefaultButtonShadowThickness"
#define XmCDefaultButtonType "DefaultButtonType"
#define XmCDefaultCopyCursorIcon "DefaultCopyCursorIcon"
#define XmCDefaultFontList "DefaultFontList"
#define XmCDefaultInvalidCursorIcon "DefaultInvalidCursorIcon"
#define XmCDefaultLinkCursorIcon "DefaultLinkCursorIcon"
#define XmCDefaultMoveCursorIcon "DefaultMoveCursorIcon"
#define XmCDefaultNoneCursorIcon "DefaultNoneCursorIcon"
#define XmCDefaultPosition "DefaultPosition"
#define XmCDefaultSourceCursorIcon "DefaultSourceCursorIcon"
#define XmCDefaultValidCursorIcon "DefaultValidCursorIcon"
#define XmCDeleteResponse "DeleteResponse"
#define XmCDesktopParent "DesktopParent"
#define XmCDialogStyle "DialogStyle"
#define XmCDialogTitle "DialogTitle"
```

6-236 LIBRARIES

Figure 6-290: <Xm/XmStrDefs.h>*, Part 3 of 34

```
#define XmCDialogType "DialogType"
#define XmCDirListItemCount "DirListItemCount"
#define XmCDirListItems "DirListItems"
#define XmCDirListLabelString "DirListLabelString"
#define XmCDirMask "DirMask"
#define XmCDirSearchProc "DirSearchProc"
#define XmCDirSpec "DirSpec"
#define XmCDirectory "Directory"
#define XmCDirectoryValid "DirectoryValid"
#define XmCDisarmCallback "DisarmCallback"
#define XmCDoubleClickInterval "DoubleClickInterval"
#define XmCDragContextClass "DragContextClass"
#define XmCDragDropFinishCallback "DragDropFinishCallback"
#define XmCDragIconClass "DragIconClass"
#define XmCDragInitiatorProtocolStyle "DragInitiatorProtocolStyle"
#define XmCDragMotionCallback "DragMotionCallback"
#define XmCDragOperations "DragOperations"
#define XmCDragOverMode "DragOverMode"
#define XmCDragProc "DragProc"
#define XmCDragReceiverProtocolStyle "DragReceiverProtocolStyle"
#define XmCDropProc "DropProc"
#define XmCDropRectangles "DropRectangles"
#define XmCDropSiteActivity "DropSiteActivity"
#define XmCDropSiteEnterCallback "DropSiteEnterCallback"
#define XmCDropSiteLeaveCallback "DropSiteLeaveCallback"
#define XmCDropSiteManagerClass "DropSiteManagerClass"
#define XmCDropSiteOperations "DropSiteOperations"
#define XmCDropSiteType "DropSiteType"
#define XmCDropStartCallback "DropStartCallback"
#define XmCDropTransferClass "DropTransferClass"
#define XmCDropTransfers "DropTransfers"
#define XmCEditable "Editable"
#define XmCEntryBorder "EntryBorder"
#define XmCEntryClass "EntryClass"
#define XmCExportTargets "ExportTargets"
```

Figure 6-291: <Xm/XmStrDefs.h>*, Part 4 of 34

```
#define XmCExposeCallback "ExposeCallback"
#define XmCExtensionType "ExtensionType"
#define XmCFileListItemCount "FileListItemCount"
#define XmCFileListItems "FileListItems"
#define XmCFileListLabelString "FileListLabelString"
#define XmCFileSearchProc "FileSearchProc"
#define XmCFileTypeMask "FileTypeMask"
#define XmCFillOnArm "FillOnArm"
#define XmCFillOnSelect "FillOnSelect"
#define XmCFilterLabelString "FilterLabelString"
#define XmCFontList "FontList"
#define XmCForegroundThreshold "ForegroundThreshold"
#define XmCHelpLabelString "HelpLabelString"
#define XmCHighlightColor "HighlightColor"
#define XmCHighlightOnEnter "HighlightOnEnter"
#define XmCHighlightPixmap "HighlightPixmap"
#define XmCHighlightThickness "HighlightThickness"
#define XmCHorizontalFontUnit "HorizontalFontUnit"
#define XmCHorizontalScrollBar "HorizontalScrollBar"
#define XmCHot "Hot"
#define XmCICCHandle "ICCHandle"
#define XmCImportTargets "ImportTargets"
#define XmCIncrement "Increment"
#define XmCIncremental "Incremental"
#define XmCIndicatorOn "IndicatorOn"
#define XmCIndicatorSize "IndicatorSize"
#define XmCIndicatorType "IndicatorType"
#define XmCInitialDelay "InitialDelay"
#define XmCInitialFocus "InitialFocus"
#define XmCInputCreate "InputCreate"
#define XmCInputMethod "InputMethod"
#define XmCInvalidCursorForeground "InvalidCursorForeground"
#define XmCIsAligned "IsAligned"
#define XmCIsHomogeneous "IsHomogeneous"
#define XmCItemCount "ItemCount"
```

6-238 LIBRARIES

Figure 6-292: <Xm/XmStrDefs.h>*, Part 5 of 34

```
#define XmCItems "Items"
#define XmCKeyboardFocusPolicy "KeyboardFocusPolicy"
#define XmCLabelFontList "LabelFontList"
#define XmCLabelInsensitivePixmap "LabelInsensitivePixmap"
#define XmCLabelPixmap "LabelPixmap"
#define XmCLabelString "LabelString"
#define XmCLabelType "LabelType"
#define XmCLightThreshold "LightThreshold"
#define XmCListLabelString "ListLabelString"
#define XmCListMarginHeight "ListMarginHeight"
#define XmCListMarginWidth "ListMarginWidth"
#define XmCListSizePolicy "ListSizePolicy"
#define XmCListSpacing "ListSpacing"
#define XmCListUpdated "ListUpdated"
#define XmCLogicalParent "LogicalParent"
#define XmCMainWindowMarginHeight "MainWindowMarginHeight"
#define XmCMainWindowMarginWidth "MainWindowMarginWidth"
#define XmCMappingDelay "MappingDelay"
#define XmCMarginBottom "MarginBottom"
#define XmCMarginHeight "MarginHeight"
#define XmCMarginLeft "MarginLeft"
#define XmCMarginRight "MarginRight"
#define XmCMarginTop "MarginTop"
#define XmCMarginWidth "MarginWidth"
#define XmCMask "Mask"
#define XmCMaxItems "MaxItems"
#define XmCMaxLength "MaxLength"
#define XmCMaxValue "MaxValue"
#define XmCMaximum "Maximum"
#define XmCMenuBar "MenuBar"
#define XmCMenuPost "MenuPost"
#define XmCMenuWidget "MenuWidget"
#define XmCMessageProc "MessageProc"
#define XmCMessageWindow "MessageWindow"
#define XmCMinimizeButtons "MinimizeButtons"
```

Figure 6-293: <Xm/XmStrDefs.h>*, Part 6 of 34

```
#define XmCMinimum "Minimum"
#define XmCMnemonic "Mnemonic"
#define XmCMnemonicCharSet "MnemonicCharSet"
#define XmCMoveOpaque "MoveOpaque"
#define XmCMultiClick "MultiClick"
#define XmCMustMatch "MustMatch"
#define XmCMwmDecorations "MwmDecorations"
#define XmCMwmFunctions "MwmFunctions"
#define XmCMwmInputMode "MwmInputMode"
#define XmCMwmMenu "MwmMenu"
#define XmCMwmMessages "MwmMessages"
#define XmCNavigationType "NavigationType"
#define XmCNeedsMotion "NeedsMotion"
#define XmCNoMatchString "NoMatchString"
#define XmCNoResize "NoResize"
#define XmCNoneCursorForeground "NoneCursorForeground"
#define XmCNotifyProc "NotifyProc"
#define XmCNumChildren "NumChildren"
#define XmCNumColumns "NumColumns"
#define XmCNumDropRectangles "NumDropRectangles"
#define XmCNumDropTransfers "NumDropTransfers"
#define XmCNumExportTargets "NumExportTargets"
#define XmCNumImportTargets "NumImportTargets"
#define XmCOffset "Offset"
#define XmCOkLabelString "OkLabelString"
#define XmCOperationChangedCallback "OperationChangedCallback"
#define XmCOperationCursorIcon "OperationCursorIcon"
#define XmCOptionLabel "OptionLabel"
#define XmCOptionMnemonic "OptionMnemonic"
#define XmCOutputCreate "OutputCreate"
#define XmCPacking "Packing"
#define XmCPageIncrement "PageIncrement"
#define XmCPaneMaximum "PaneMaximum"
#define XmCPaneMinimum "PaneMinimum"
#define XmCPattern "Pattern"
```

6-240 LIBRARIES

Figure 6-294: <Xm/XmStrDefs.h>*, Part 7 of 34

```
#define XmCPendingDelete "PendingDelete"
#define XmCPopupEnabled "PopupEnabled"
#define XmCPositionIndex "PositionIndex"
#define XmCPostFromButton "PostFromButton"
#define XmCPostFromCount "PostFromCount"
#define XmCPostFromList "PostFromList"
#define XmCPreeditType "PreeditType"
#define XmCProcessingDirection "ProcessingDirection"
#define XmCPromptString "PromptString"
#define XmCProtocolCallback "ProtocolCallback"
#define XmCPushButtonEnabled "PushButtonEnabled"
#define XmCQualifySearchDataProc "QualifySearchDataProc"
#define XmCRadioAlwaysOne "RadioAlwaysOne"
#define XmCRadioBehavior "RadioBehavior"
#define XmCRecomputeSize "RecomputeSize"
#define XmCRectangles "Rectangles"
#define XmCRepeatDelay "RepeatDelay"
#define XmCResizeCallback "ResizeCallback"
#define XmCResizeHeight "ResizeHeight"
#define XmCResizePolicy "ResizePolicy"
#define XmCResizeWidth "ResizeWidth"
#define XmCRowColumnType "RowColumnType"
#define XmCRows "Rows"
#define XmCRubberPositioning "RubberPositioning"
#define XmCSashHeight "SashHeight"
#define XmCSashIndent "SashIndent"
#define XmCSashWidth "SashWidth"
#define XmCScaleHeight "ScaleHeight"
#define XmCScaleMultiple "ScaleMultiple"
#define XmCScaleWidth "ScaleWidth"
#define XmCScroll "Scroll"
#define XmCScrollBarDisplayPolicy "ScrollBarDisplayPolicy"
#define XmCScrollBarPlacement "ScrollBarPlacement"
#define XmCScrollSide "ScrollSide"
#define XmCScrolledWindowMarginHeight "ScrolledWindowMarginHeight"
```

Figure 6-295: <Xm/XmStrDefs.h>*, Part 8 of 34

```
#define XmCScrolledWindowMarginWidth "ScrolledWindowMarginWidth"
#define XmCScrollingPolicy "ScrollingPolicy"
#define XmCSelectColor "SelectColor"
#define XmCSelectInsensitivePixmap "SelectInsensitivePixmap"
#define XmCSelectPixmap "SelectPixmap"
#define XmCSelectThreshold "SelectThreshold"
#define XmCSelectedItemCount "SelectedItemCount"
#define XmCSelectedItems "SelectedItems"
#define XmCSelectionArrayCount "SelectionArrayCount"
#define XmCSelectionLabelString "SelectionLabelString"
#define XmCSelectionPolicy "SelectionPolicy"
#define XmCSeparatorOn "SeparatorOn"
#define XmCSeparatorType "SeparatorType"
#define XmCSet "Set"
#define XmCShadowThickness "ShadowThickness"
#define XmCShadowType "ShadowType"
#define XmCShellUnitType "ShellUnitType"
#define XmCShowArrows "ShowArrows"
#define XmCShowAsDefault "ShowAsDefault"
#define XmCShowSeparator "ShowSeparator"
#define XmCShowValue "ShowValue"
#define XmCSimpleCheckBox "SimpleCheckBox"
#define XmCSimpleMenuBar "SimpleMenuBar"
#define XmCSimpleOptionMenu "SimpleOptionMenu"
#define XmCSimplePopupMenu "SimplePopupMenu"
#define XmCSimplePulldownMenu "SimplePulldownMenu"
#define XmCSimpleRadioBox "SimpleRadioBox"
#define XmCSizePolicy "SizePolicy"
#define XmCSliderSize "SliderSize"
#define XmCSource "Source"
#define XmCSourceCursorIcon "SourceCursorIcon"
#define XmCSourceIsExternal "SourceIsExternal"
#define XmCSourcePixmapIcon "SourcePixmapIcon"
#define XmCSourceWidget "SourceWidget"
#define XmCSourceWindow "SourceWindow"
```

6-242 LIBRARIES

Figure 6-296: <Xm/XmStrDefs.h>*, Part 9 of 34

```
#define XmCSpacing "Spacing"
#define XmCStartTime "StartTime"
#define XmCStateCursorIcon "StateCursorIcon"
#define XmCStringDirection "StringDirection"
#define XmCTearOffModel "TearOffModel"
#define XmCTextFontList "TextFontList"
#define XmCTextString "TextString"
#define XmCTextValue "TextValue"
#define XmCTitleString "TitleString"
#define XmCTopCharacter "TopCharacter"
#define XmCTopItemPosition "TopItemPosition"
#define XmCTopLevelEnterCallback "TopLevelEnterCallback"
#define XmCTopLevelLeaveCallback "TopLevelLeaveCallback"
#define XmCTopShadowColor "TopShadowColor"
#define XmCTopShadowPixmap "TopShadowPixmap"
#define XmCTransferProc "TransferProc"
#define XmCTransferStatus "TransferStatus"
#define XmCTraversalOn "TraversalOn"
#define XmCTraversalType "TraversalType"
#define XmCTreeUpdateProc "TreeUpdateProc"
#define XmCTroughColor "TroughColor"
#define XmCUnitType "UnitType"
#define XmCUnpostBehavior "UnpostBehavior"
#define XmCUnselectPixmap "UnselectPixmap"
#define XmCUpdateSliderSize "UpdateSliderSize"
#define XmCUseAsyncGeometry "UseAsyncGeometry"
#define XmCUserData "UserData"
#define XmCValidCursorForeground "ValidCursorForeground"
#define XmCValueChangedCallback "ValueChangedCallback"
#define XmCValueWcs "ValueWcs"
#define XmCVerifyBell "VerifyBell"
#define XmCVerticalAlignment "VerticalAlignment"
#define XmCVerticalFontUnit "VerticalFontUnit"
#define XmCVerticalScrollBar "VerticalScrollBar"
```

Figure 6-297: <Xm/XmStrDefs.h>*, Part 10 of 34

```
#define XmCVisibleItemCount "VisibleItemCount"
#define XmCVisibleWhenOff "VisibleWhenOff"
#define XmCVisualPolicy "VisualPolicy"
#define XmCWhichButton "WhichButton"
#define XmCWordWrap "WordWrap"
#define XmCWorkWindow "WorkWindow"
#define XmCXmString "XmString"
#define XmNaccelerator "accelerator"
#define XmNacceleratorText "acceleratorText"
#define XmNactivateCallback "activateCallback"
#define XmNadjustLast "adjustLast"
#define XmNadjustMargin "adjustMargin"
#define XmNalignment "alignment"
#define XmNallowOverlap "allowOverlap"
#define XmNallowResize "allowResize"
#define XmNanimationMask "animationMask"
#define XmNanimationPixmap "animationPixmap"
#define XmNanimationPixmapDepth "animationPixmapDepth"
#define XmNanimationStyle "animationStyle"
#define XmNapplyCallback "applyCallback"
#define XmNapplyLabelString "applyLabelString"
#define XmNarmCallback "armCallback"
#define XmNarmColor "armColor"
#define XmNarmPixmap "armPixmap"
#define XmNarrowDirection "arrowDirection"
#define XmNattachment "attachment"
#define XmNaudibleWarning "audibleWarning"
#define XmNautoShowCursorPosition "autoShowCursorPosition"
#define XmNautoUnmanage "autoUnmanage"
#define XmNautomaticSelection "automaticSelection"
#define XmNavailability "availability"
#define XmNblendModel "blendModel"
#define XmNblinkRate "blinkRate"
#define XmNbottomAttachment "bottomAttachment"
#define XmNbottomOffset "bottomOffset"
```

6-244 LIBRARIES

Figure 6-298: <Xm/XmStrDefs.h>*, Part 11 of 34

```
#define XmNbottomPosition "bottomPosition"
#define XmNbottomShadowColor "bottomShadowColor"
#define XmNbottomShadowPixmap "bottomShadowPixmap"
#define XmNbottomWidget "bottomWidget"
#define XmNbrowseSelectionCallback "browseSelectionCallback"
#define XmNbuttonAcceleratorText "buttonAcceleratorText"
#define XmNbuttonAccelerators "buttonAccelerators"
#define XmNbuttonCount "buttonCount"
#define XmNbuttonFontList "buttonFontList"
#define XmNbuttonMnemonicCharSets "buttonMnemonicCharSets"
#define XmNbuttonMnemonics "buttonMnemonics"
#define XmNbuttonSet "buttonSet"
#define XmNbuttonType "buttonType"
#define XmNbuttons "buttons"
#define XmNcancelButton "cancelButton"
#define XmNcancelCallback "cancelCallback"
#define XmNcancelLabelString "cancelLabelString"
#define XmNcascadePixmap "cascadePixmap"
#define XmNcascadingCallback "cascadingCallback"
#define XmNchildHorizontalAlignment "childHorizontalAlignment"
#define XmNchildHorizontalSpacing "childHorizontalSpacing"
#define XmNchildPlacement "childPlacement"
#define XmNchildPosition "childPosition"
#define XmNchildType "childType"
#define XmNchildVerticalAlignment "childVerticalAlignment"
#define XmNclientData "clientData"
#define XmNclipWindow "clipWindow"
#define XmNcolumns "columns"
#define XmNcommand "command"
#define XmNcommandChangedCallback "commandChangedCallback"
#define XmNcommandEnteredCallback "commandEnteredCallback"
#define XmNcommandWindow "commandWindow"
#define XmNcommandWindowLocation "commandWindowLocation"
#define XmNconvertProc "convertProc"
#define XmNcursorBackground "cursorBackground"
```

Figure 6-299: <Xm/XmStrDefs.h>*, Part 12 of 34

```
#define XmNcursorForeground "cursorForeground"
#define XmNcursorPosition "cursorPosition"
#define XmNcursorPositionVisible "cursorPositionVisible"
#define XmNdarkThreshold "darkThreshold"
#define XmNdecimalPoints "decimalPoints"
#define XmNdecrementCallback "decrementCallback"
#define XmNdefaultActionCallback "defaultActionCallback"
#define XmNdefaultButton "defaultButton"
#define XmNdefaultButtonShadowThickness "defaultButtonShadowThickness"
#define XmNdefaultButtonType "defaultButtonType"
#define XmNdefaultCopyCursorIcon "defaultCopyCursorIcon"
#define XmNdefaultFontList "defaultFontList"
#define XmNdefaultInvalidCursorIcon "defaultInvalidCursorIcon"
#define XmNdefaultLinkCursorIcon "defaultLinkCursorIcon"
#define XmNdefaultMoveCursorIcon "defaultMoveCursorIcon"
#define XmNdefaultNoneCursorIcon "defaultNoneCursorIcon"
#define XmNdefaultPosition "defaultPosition"
#define XmNdefaultSourceCursorIcon "defaultSourceCursorIcon"
#define XmNdefaultValidCursorIcon "defaultValidCursorIcon"
#define XmNdeleteResponse "deleteResponse"
#define XmNdesktopParent "desktopParent"
#define XmNdialogStyle "dialogStyle"
#define XmNdialogTitle "dialogTitle"
#define XmNdialogType "dialogType"
#define XmNdirListItemCount "dirListItemCount"
#define XmNdirListItems "dirListItems"
#define XmNdirListLabelString "dirListLabelString"
#define XmNdirMask "dirMask"
#define XmNdirSearchProc "dirSearchProc"
#define XmNdirSpec "dirSpec"
#define XmNdirectory "directory"
#define XmNdirectoryValid "directoryValid"
#define XmNdisarmCallback "disarmCallback"
#define XmNdoubleClickInterval "doubleClickInterval"
#define XmNdragCallback "dragCallback"
```

6-246 LIBRARIES

Figure 6-300: <Xm/XmStrDefs.h>*, Part 13 of 34

```
#define XmNdragContextClass "dragContextClass"
#define XmNdragDropFinishCallback "dragDropFinishCallback"
#define XmNdragIconClass "dragIconClass"
#define XmNdragInitiatorProtocolStyle "dragInitiatorProtocolStyle"
#define XmNdragMotionCallback "dragMotionCallback"
#define XmNdragOperations "dragOperations"
#define XmNdragOverMode "dragOverMode"
#define XmNdragProc "dragProc"
#define XmNdragReceiverProtocolStyle "dragReceiverProtocolStyle"
#define XmNdropFinishCallback "dropFinishCallback"
#define XmNdropProc "dropProc"
#define XmNdropRectangles "dropRectangles"
#define XmNdropSiteActivity "dropSiteActivity"
#define XmNdropSiteEnterCallback "dropSiteEnterCallback"
#define XmNdropSiteLeaveCallback "dropSiteLeaveCallback"
#define XmNdropSiteManagerClass "dropSiteManagerClass"
#define XmNdropSiteOperations "dropSiteOperations"
#define XmNdropSiteType "dropSiteType"
#define XmNdropStartCallback "dropStartCallback"
#define XmNdropTransferClass "dropTransferClass"
#define XmNdropTransfers "dropTransfers"
#define XmNeditMode "editMode"
#define XmNeditable "editable"
#define XmNentryAlignment "entryAlignment"
#define XmNentryBorder "entryBorder"
#define XmNentryCallback "entryCallback"
#define XmNentryClass "entryClass"
#define XmNentryVerticalAlignment "entryVerticalAlignment"
#define XmNexportTargets "exportTargets"
#define XmNexposeCallback "exposeCallback"
#define XmNextendedSelectionCallback "extendedSelectionCallback"
#define XmNextensionType "extensionType"
#define XmNfileListItemCount "fileListItemCount"
#define XmNfileListItems "fileListItems"
#define XmNfileListLabelString "fileListLabelString"
```

Figure 6-301: <Xm/XmStrDefs.h>*, Part 14 of 34

```
#define XmNfileSearchProc "fileSearchProc"
#define XmNfileTypeMask "fileTypeMask"
#define XmNfillOnArm "fillOnArm"
#define XmNfillOnSelect "fillOnSelect"
#define XmNfilterLabelString "filterLabelString"
#define XmNfocusCallback "focusCallback"
#define XmNfocusMovedCallback "focusMovedCallback"
#define XmNfocusPolicyChanged "focusPolicyChanged"
#define XmNfontList "fontList"
#define XmNforegroundThreshold "foregroundThreshold"
#define XmNfractionBase "fractionBase"
#define XmNgainPrimaryCallback "gainPrimaryCallback"
#define XmNhelpCallback "helpCallback"
#define XmNhelpLabelString "helpLabelString"
#define XmNhighlightColor "highlightColor"
#define XmNhighlightOnEnter "highlightOnEnter"
#define XmNhighlightPixmap "highlightPixmap"
#define XmNhighlightThickness "highlightThickness"
#define XmNhistoryItemCount "historyItemCount"
#define XmNhistoryItems "historyItems"
#define XmNhistoryMaxItems "historyMaxItems"
#define XmNhistoryVisibleItemCount "historyVisibleItemCount"
#define XmNhorizontalFontUnit "horizontalFontUnit"
#define XmNhorizontalScrollBar "horizontalScrollBar"
#define XmNhorizontalSpacing "horizontalSpacing"
#define XmNhotX "hotX"
#define XmNhotY "hotY"
#define XmNiccHandle "iccHandle"
#define XmNimportTargets "importTargets"
#define XmNincrement "increment"
#define XmNincrementCallback "incrementCallback"
#define XmNincremental "incremental"
#define XmNindicatorOn "indicatorOn"
#define XmNindicatorSize "indicatorSize"
#define XmNindicatorType "indicatorType"
```

6-248 LIBRARIES

Figure 6-302: <Xm/XmStrDefs.h>*, Part 15 of 34

```
#define XmNinitialDelay "initialDelay"
#define XmNinitialFocus "initialFocus"
#define XmNinputCallback "inputCallback"
#define XmNinputCreate "inputCreate"
#define XmNinputMethod "inputMethod"
#define XmNinvalidCursorForeground "invalidCursorForeground"
#define XmNisAligned "isAligned"
#define XmNisHomogeneous "isHomogeneous"
#define XmNitemCount "itemCount"
#define XmNitems "items"
#define XmNkeyboardFocusPolicy "keyboardFocusPolicy"
#define XmNlabelFontList "labelFontList"
#define XmNlabelInsensitivePixmap "labelInsensitivePixmap"
#define XmNlabelPixmap "labelPixmap"
#define XmNlabelString "labelString"
#define XmNlabelType "labelType"
#define XmNleftAttachment "leftAttachment"
#define XmNleftOffset "leftOffset"
#define XmNleftPosition "leftPosition"
#define XmNleftWidget "leftWidget"
#define XmNlightThreshold "lightThreshold"
#define XmNlineSpace "lineSpace"
#define XmNlistItemCount "listItemCount"
#define XmNlistItems "listItems"
#define XmNlistLabelString "listLabelString"
#define XmNlistMarginHeight "listMarginHeight"
#define XmNlistMarginWidth "listMarginWidth"
#define XmNlistSizePolicy "listSizePolicy"
#define XmNlistSpacing "listSpacing"
#define XmNlistUpdated "listUpdated"
#define XmNlistVisibleItemCount "listVisibleItemCount"
#define XmNlogicalParent "logicalParent"
#define XmNlosePrimaryCallback "losePrimaryCallback"
#define XmNlosingFocusCallback "losingFocusCallback"
#define XmNmainWindowMarginHeight "mainWindowMarginHeight"
```

Figure 6-303: <Xm/XmStrDefs.h>*, Part 16 of 34

```
#define XmNmainWindowMarginWidth "mainWindowMarginWidth"
#define XmNmapCallback "mapCallback"
#define XmNmappingDelay "mappingDelay"
#define XmNmargin "margin"
#define XmNmarginBottom "marginBottom"
#define XmNmarginHeight "marginHeight"
#define XmNmarginLeft "marginLeft"
#define XmNmarginRight "marginRight"
#define XmNmarginTop "marginTop"
#define XmNmarginWidth "marginWidth"
#define XmNmask "mask"
#define XmNmaxLength "maxLength"
#define XmNmaximum "maximum"
#define XmNmenuAccelerator "menuAccelerator"
#define XmNmenuBar "menuBar"
#define XmNmenuCursor "menuCursor"
#define XmNmenuHelpWidget "menuHelpWidget"
#define XmNmenuHistory "menuHistory"
#define XmNmenuPost "menuPost"
#define XmNmessageAlignment "messageAlignment"
#define XmNmessageProc "messageProc"
#define XmNmessageString "messageString"
#define XmNmessageWindow "messageWindow"
#define XmNminimizeButtons "minimizeButtons"
#define XmNminimum "minimum"
#define XmNmnemonic "mnemonic"
#define XmNmnemonicCharSet "mnemonicCharSet"
#define XmNmodifyVerifyCallback "modifyVerifyCallback"
#define XmNmodifyVerifyCallbackWcs "modifyVerifyCallbackWcs"
#define XmNmotionVerifyCallback "motionVerifyCallback"
#define XmNmoveOpaque "moveOpaque"
#define XmNmultiClick "multiClick"
#define XmNmultipleSelectionCallback "multipleSelectionCallback"
#define XmNmustMatch "mustMatch"
#define XmNmwmDecorations "mwmDecorations"
```

6-250 LIBRARIES

Figure 6-304: <Xm/XmStrDefs.h>*, Part 17 of 34

```
#define XmNmwmFunctions "mwmFunctions"
#define XmNmwmInputMode "mwmInputMode"
#define XmNmwmMenu "mwmMenu"
#define XmNmwmMessages "mwmMessages"
#define XmNnavigationType "navigationType"
#define XmNneedsMotion "needsMotion"
#define XmNnoMatchCallback "noMatchCallback"
#define XmNnoMatchString "noMatchString"
#define XmNnoResize "noResize"
#define XmNnoneCursorForeground "noneCursorForeground"
#define XmNnotifyProc "notifyProc"
#define XmNnumColumns "numColumns"
#define XmNnumDropRectangles "numDropRectangles"
#define XmNnumDropTransfers "numDropTransfers"
#define XmNnumExportTargets "numExportTargets"
#define XmNnumImportTargets "numImportTargets"
#define XmNnumRectangles "numRectangles"
#define XmNoffsetX "offsetX"
#define XmNoffsetY "offsetY"
#define XmNokCallback "okCallback"
#define XmNokLabelString "okLabelString"
#define XmNoperationChangedCallback "operationChangedCallback"
#define XmNoperationCursorIcon "operationCursorIcon"
#define XmNoptionLabel "optionLabel"
#define XmNoptionMnemonic "optionMnemonic"
#define XmNoutputCreate "outputCreate"
#define XmNpacking "packing"
#define XmNpageDecrementCallback "pageDecrementCallback"
#define XmNpageIncrement "pageIncrement"
#define XmNpageIncrementCallback "pageIncrementCallback"
#define XmNpaneMaximum "paneMaximum"
#define XmNpaneMinimum "paneMinimum"
#define XmNpattern "pattern"
#define XmNpendingDelete "pendingDelete"
#define XmNpopupEnabled "popupEnabled"
```

Figure 6-305: <Xm/XmStrDefs.h>*, Part 18 of 34

```
#define XmNpositionIndex "positionIndex"
#define XmNpostFromButton "postFromButton"
#define XmNpostFromCount "postFromCount"
#define XmNpostFromList "postFromList"
#define XmNpreeditType "preeditType"
#define XmNprocessingDirection "processingDirection"
#define XmNpromptString "promptString"
#define XmNprotocolCallback "protocolCallback"
#define XmNpushButtonEnabled "pushButtonEnabled"
#define XmNqualifySearchDataProc "qualifySearchDataProc"
#define XmNradioAlwaysOne "radioAlwaysOne"
#define XmNradioBehavior "radioBehavior"
#define XmNrealizeCallback "realizeCallback"
#define XmNrecomputeSize "recomputeSize"
#define XmNrectangles "rectangles"
#define XmNrefigureMode "refigureMode"
#define XmNrepeatDelay "repeatDelay"
#define XmNresizable "resizable"
#define XmNresizeCallback "resizeCallback"
#define XmNresizeHeight "resizeHeight"
#define XmNresizePolicy "resizePolicy"
#define XmNresizeWidth "resizeWidth"
#define XmNrightAttachment "rightAttachment"
#define XmNrightOffset "rightOffset"
#define XmNrightPosition "rightPosition"
#define XmNrightWidget "rightWidget"
#define XmNrowColumnType "rowColumnType"
#define XmNrows "rows"
#define XmNrubberPositioning "rubberPositioning"
#define XmNsashHeight "sashHeight"
#define XmNsashIndent "sashIndent"
#define XmNsashShadowThickness "sashShadowThickness"
#define XmNsashWidth "sashWidth"
#define XmNscaleHeight "scaleHeight"
#define XmNscaleMultiple "scaleMultiple"
```

6-252 LIBRARIES

Figure 6-306: <Xm/XmStrDefs.h>*, Part 19 of 34

```
#define XmNscaleWidth "scaleWidth"
#define XmNscrollBarDisplayPolicy "scrollBarDisplayPolicy"
#define XmNscrollBarPlacement "scrollBarPlacement"
#define XmNscrollHorizontal "scrollHorizontal"
#define XmNscrollLeftSide "scrollLeftSide"
#define XmNscrollTopSide "scrollTopSide"
#define XmNscrollVertical "scrollVertical"
#define XmNscrolledWindowMarginHeight "scrolledWindowMarginHeight"
#define XmNscrolledWindowMarginWidth "scrolledWindowMarginWidth"
#define XmNscrollingPolicy "scrollingPolicy"
#define XmNselectColor "selectColor"
#define XmNselectInsensitivePixmap "selectInsensitivePixmap"
#define XmNselectPixmap "selectPixmap"
#define XmNselectThreshold "selectThreshold"
#define XmNselectedItemCount "selectedItemCount"
#define XmNselectedItems "selectedItems"
#define XmNselectionArrayCount "selectionArrayCount"
#define XmNselectionLabelString "selectionLabelString"
#define XmNselectionPolicy "selectionPolicy"
#define XmNseparatorOn "separatorOn"
#define XmNseparatorType "separatorType"
#define XmNset "set"
#define XmNshadow "shadow"
#define XmNshadowThickness "shadowThickness"
#define XmNshadowType "shadowType"
#define XmNshellUnitType "shellUnitType"
#define XmNshowArrows "showArrows"
#define XmNshowAsDefault "showAsDefault"
#define XmNshowSeparator "showSeparator"
#define XmNshowValue "showValue"
#define XmNsimpleCallback "simpleCallback"
#define XmNsingleSelectionCallback "singleSelectionCallback"
#define XmNsizePolicy "sizePolicy"
#define XmNskipAdjust "skipAdjust"
#define XmNsliderSize "sliderSize"
```

Figure 6-307: <Xm/XmStrDefs.h>*, Part 20 of 34

```
#define XmNsource "source"
#define XmNsourceCursorIcon "sourceCursorIcon"
#define XmNsourceIsExternal "sourceIsExternal"
#define XmNsourcePixmapIcon "sourcePixmapIcon"
#define XmNsourceWidget "sourceWidget"
#define XmNsourceWindow "sourceWindow"
#define XmNspacing "spacing"
#define XmNspotLocation "spotLocation"
#define XmNstartTime "startTime"
#define XmNstateCursorIcon "stateCursorIcon"
#define XmNstringDirection "stringDirection"
#define XmNsubMenuId "subMenuId"
#define XmNsymbolPixmap "symbolPixmap"
#define XmNtearOffMenuActivateCallback "tearOffMenuActivateCallback"
#define XmNtearOffMenuDeactivateCallback "tearOffMenuDeactivateCallback"
#define XmNtearOffModel "tearOffModel"
#define XmNtextAccelerators "textAccelerators"
#define XmNtextColumns "textColumns"
#define XmNtextFontList "textFontList"
#define XmNtextString "textString"
#define XmNtextTranslations "textTranslations"
#define XmNtextValue "textValue"
#define XmNtitleString "titleString"
#define XmNtoBottomCallback "toBottomCallback"
#define XmNtoPositionCallback "toPositionCallback"
#define XmNtoTopCallback "toTopCallback"
#define XmNtopAttachment "topAttachment"
#define XmNtopCharacter "topCharacter"
#define XmNtopItemPosition "topItemPosition"
#define XmNtopLevelEnterCallback "topLevelEnterCallback"
#define XmNtopLevelLeaveCallback "topLevelLeaveCallback"
#define XmNtopOffset "topOffset"
#define XmNtopPosition "topPosition"
#define XmNtopShadowColor "topShadowColor"
#define XmNtopShadowPixmap "topShadowPixmap"
```

6-254 LIBRARIES

Figure 6-308: <Xm/XmStrDefs.h>*, Part 21 of 34

```
#define XmNtopWidget "topWidget"
#define XmNtransferProc "transferProc"
#define XmNtransferStatus "transferStatus"
#define XmNtraversalCallback "traversalCallback"
#define XmNtraversalOn "traversalOn"
#define XmNtraversalType "traversalType"
#define XmNtraverseObscuredCallback "traverseObscuredCallback"
#define XmNtreeUpdateProc "treeUpdateProc"
#define XmNtroughColor "troughColor"
#define XmNunitType "unitType"
#define XmNunmapCallback "unmapCallback"
#define XmNunpostBehavior "unpostBehavior"
#define XmNunselectPixmap "unselectPixmap"
#define XmNupdateSliderSize "updateSliderSize"
#define XmNuseAsyncGeometry "useAsyncGeometry"
#define XmNuserData "userData"
#define XmNvalidCursorForeground "validCursorForeground"
#define XmNvalueChangedCallback "valueChangedCallback"
#define XmNvalueWcs "valueWcs"
#define XmNverifyBell "verifyBell"
#define XmNverticalFontUnit "verticalFontUnit"
#define XmNverticalScrollBar "verticalScrollBar"
#define XmNverticalSpacing "verticalSpacing"
#define XmNvisibleItemCount "visibleItemCount"
#define XmNvisibleWhenOff "visibleWhenOff"
#define XmNvisualPolicy "visualPolicy"
#define XmNwhichButton "whichButton"
#define XmNwordWrap "wordWrap"
#define XmNworkWindow "workWindow"
#define XmRAlignment "Alignment"
#define XmRAnimationMask "AnimationMask"
#define XmRAnimationPixmap "AnimationPixmap"
#define XmRAnimationStyle "AnimationStyle"
#define XmRArrowDirection "ArrowDirection"
#define XmRAtomList "AtomList"
```

Figure 6-309: <Xm/XmStrDefs.h>*, Part 22 of 34

```
#define XmRAttachment "Attachment"
#define XmRAudibleWarning "AudibleWarning"
#define XmRAvailability "Availability"
#define XmRBackgroundPixmap "BackgroundPixmap"
#define XmRBlendModel "BlendModel"
#define XmRBooleanDimension "BooleanDimension"
#define XmRBottomShadowPixmap "BottomShadowPixmap"
#define XmRButtonType "ButtonType"
#define XmRCallbackProc "CallbackProc"
#define XmRChar "Char"
#define XmRCharSetTable "CharSetTable"
#define XmRChildHorizontalAlignment "ChildHorizontalAlignment"
#define XmRChildPlacement "ChildPlacement"
#define XmRChildType "ChildType"
#define XmRChildVerticalAlignment "ChildVerticalAlignment"
#define XmRCommandWindowLocation "CommandWindowLocation"
#define XmRCompoundText "CompoundText"
#define XmRDefaultButtonType "DefaultButtonType"
#define XmRDeleteResponse "DeleteResponse"
#define XmRDialogStyle "DialogStyle"
#define XmRDialogType "DialogType"
#define XmRDoubleClickInterval "DoubleClickInterval"
#define XmRDragInitiatorProtocolStyle "DragInitiatorProtocolStyle"
#define XmRDragReceiverProtocolStyle "DragReceiverProtocolStyle"
#define XmRDropSiteActivity "DropSiteActivity"
#define XmRDropSiteOperations "DropSiteOperations"
#define XmRDropSiteType "DropSiteType"
#define XmRDropTransfers "DropTransfers"
#define XmRExtensionType "ExtensionType"
#define XmRFileTypeMask "FileTypeMask"
#define XmRFontList "FontList"
#define XmRGadgetPixmap "GadgetPixmap"
#define XmRHighlightPixmap "HighlightPixmap"
#define XmRHorizontalDimension "HorizontalDimension"
#define XmRHorizontalInt "HorizontalInt"
```

6-256 LIBRARIES

Figure 6-310: <Xm/XmStrDefs.h>*, Part 23 of 34

```
#define XmRHorizontalPosition "HorizontalPosition"
#define XmRIconAttachment "IconAttachment"
#define XmRImportTargets "ImportTargets"
#define XmRIndicatorType "IndicatorType"
#define XmRItemCount "ItemCount"
#define XmRItems "Items"
#define XmRKeySym "KeySym"
#define XmRKeySymTable "KeySymTable"
#define XmRKeyboardFocusPolicy "KeyboardFocusPolicy"
#define XmRLabelType "LabelType"
#define XmRListMarginHeight "ListMarginHeight"
#define XmRListMarginWidth "ListMarginWidth"
#define XmRListSizePolicy "ListSizePolicy"
#define XmRListSpacing "ListSpacing"
#define XmRManBottomShadowPixmap "ManBottomShadowPixmap"
#define XmRManForegroundPixmap "ManForegroundPixmap"
#define XmRManHighlightPixmap "ManHighlightPixmap"
#define XmRManTopShadowPixmap "ManTopShadowPixmap"
#define XmRMenuWidget "MenuWidget"
#define XmRMnemonic "Mnemonic"
#define XmRMultiClick "MultiClick"
#define XmRNavigationType "NavigationType"
#define XmRPacking "Packing"
#define XmRPrimForegroundPixmap "PrimForegroundPixmap"
#define XmRProc "Proc"
#define XmRProcessingDirection "ProcessingDirection"
#define XmRRectangleList "RectangleList"
#define XmRResizePolicy "ResizePolicy"
#define XmRRowColumnType "RowColumnType"
#define XmRScrollBarDisplayPolicy "ScrollBarDisplayPolicy"
#define XmRScrollBarPlacement "ScrollBarPlacement"
#define XmRScrollingPolicy "ScrollingPolicy"
#define XmRSelectedItemCount "SelectedItemCount"
#define XmRSelectedItems "SelectedItems"
#define XmRSelectionPolicy "SelectionPolicy"
```

Figure 6-311: <Xm/XmStrDefs.h>*, Part 24 of 34

```
#define XmRSelectionType "SelectionType"
#define XmRSeparatorType "SeparatorType"
#define XmRShadowType "ShadowType"
#define XmRShellHorizDim "ShellHorizDim"
#define XmRShellHorizPos "ShellHorizPos"
#define XmRShellUnitType "ShellUnitType"
#define XmRShellVertDim "ShellVertDim"
#define XmRShellVertPos "ShellVertPos"
#define XmRSizePolicy "SizePolicy"
#define XmRStringDirection "StringDirection"
#define XmRTearOffModel "TearOffModel"
#define XmRTopShadowPixmap "TopShadowPixmap"
#define XmRTransferStatus "TransferStatus"
#define XmRTraversalType "TraversalType"
#define XmRUnitType "UnitType"
#define XmRUnpostBehavior "UnpostBehavior"
#define XmRValueWcs "ValueWcs"
#define XmRVerticalAlignment "VerticalAlignment"
#define XmRVerticalDimension "VerticalDimension"
#define XmRVerticalInt "VerticalInt"
#define XmRVerticalPosition "VerticalPosition"
#define XmRVirtualBinding "VirtualBinding"
#define XmRVisibleItemCount "VisibleItemCount"
#define XmRVisualPolicy "VisualPolicy"
#define XmRWhichButton "WhichButton"
#define XmRXmBackgroundPixmap "XmBackgroundPixmap"
#define XmRXmString "XmString"
#define XmRXmStringCharSet "XmStringCharSet"
#define XmRXmStringTable "XmStringTable"
#define XmVosfActivate "osfActivate"
#define XmVosfAddMode "osfAddMode"
#define XmVosfBackSpace "osfBackSpace"
#define XmVosfBeginLine "osfBeginLine"
#define XmVosfCancel "osfCancel"
#define XmVosfClear "osfClear"
```

6-258 LIBRARIES

Figure 6-312: <Xm/XmStrDefs.h>*, Part 25 of 34

```
#define XmVosfCopy "osfCopy"
#define XmVosfCut "osfCut"
#define XmVosfDelete "osfDelete"
#define XmVosfDown "osfDown"
#define XmVosfEndLine "osfEndLine"
#define XmVosfHelp "osfHelp"
#define XmVosfInsert "osfInsert"
#define XmVosfLeft "osfLeft"
#define XmVosfMenu "osfMenu"
#define XmVosfMenuBar "osfMenuBar"
#define XmVosfPageDown "osfPageDown"
#define XmVosfPageLeft "osfPageLeft"
#define XmVosfPageRight "osfPageRight"
#define XmVosfPageUp "osfPageUp"
#define XmVosfPaste "osfPaste"
#define XmVosfPrimaryPaste "osfPrimaryPaste"
#define XmVosfQuickPaste "osfQuickPaste"
#define XmVosfRight "osfRight"
#define XmVosfSelect "osfSelect"
#define XmVosfUndo "osfUndo"
#define XmVosfUp "osfUp"
#define XmSFONTLIST_DEFAULT_TAG_STRING "FONTLIST_DEFAULT_TAG_STRING"
#define XmSXmFONTLIST DEFAULT TAG STRING "XmFONTLIST DEFAULT TAG STRING"
#define _XmConst /**/
#define XmSTRING_DEFAULT_CHARSET
                                        XmS
#define XmSTRING ISO8859 1
                                        "ISO8859-1"
#define XmFONTLIST DEFAULT TAG
                                        XmSFONTLIST DEFAULT TAG STRING
#define XmFONTLIST_DEFAULT_TAG_STRING XmSXmFONTLIST_DEFAULT_TAG_STRING
#define XmVaCASCADEBUTTON
                                        "cascadeButton"
#define XmVaCHECKBUTTON
                                        "checkButton"
#define XmVaDOUBLE_SEPARATOR
                                        "doubleSeparator"
#define XmVaPUSHBUTTON
                                        "pushButton"
#define XmVaRADIOBUTTON
                                        "radioButton"
```

6-259

Figure 6-313: <Xm/XmStrDefs.h>*, Part 26 of 34

#define	XmVaSEPARATOR XmVaSINGLE_SEPARATOR XmVaTOGGLEBUTTON	"separator" "singleSeparator" "checkButton"
#define	XmVaTITLE	XtNtitle
#define	XtCKeyboardFocusPolicy	XmCKeyboardFocusPolicy
#define	XtCShellUnitType	XmCShellUnitType
#define	XtNkeyboardFocusPolicy	XmNkeyboardFocusPolicy
#define	XtNshellUnitType	XmNshellUnitType
#define	XtRKeyboardFocusPolicy	XmRKeyboardFocusPolicy
	XmRPrimBottomShadowPixmap	XmRBottomShadowPixmap
	XmRPrimHighlightPixmap	XmRHighlightPixmap
#define 	XmRPrimTopShadowPixmap	XmRTopShadowPixmap
"	XmCAccelerators	XtCAccelerators
#define	XmCAllowShellResize	XtCAllowShellResize
#define	2	XtCArgc
#define		XtCArgv
	XmCBackground	XtCBackground
	XmCBaseHeight	XtCBaseHeight
	XmCBaseHeight	XtCBaseHeight
#define	XmCBaseWidth	XtCBaseWidth
	XmCBaseWidth	XtCBaseWidth
#define	XmCBitmap	XtCBitmap
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	XmCBoolean	XtCBoolean
#define	XmCBorderColor	XtCBorderColor
#define	XmCBorderWidth	XtCBorderWidth
	XmCCallback	XtCCallback
#define	XmCColor	XtCColor
	XmCColormap	XtCColormap
#define	XmCCreatePopupChildProc	XtCCreatePopupChildProc
#define	XmCCursor	XtCCursor
l	XmCDepth	XtCDepth
#define	XmCDimension	XtRDimension

6-260 LIBRARIES

Figure 6-314: <Xm/XmStrDefs.h>*, Part 27 of 34

#define	XmCEditMode	XtREditMode
#define	XmCEditType	XtCEditType
#define	XmCEventBindings	XtCEventBindings
#define	XmCFile	XtCFile
#define	XmCFont	XtCFont
#define	XmCFontSet	XtCFontSet
#define	XmCForeground	XtCForeground
#define	XmCFraction	XtCFraction
#define	XmCFunction	XtCFunction
#define	XmCGeometry	XtCGeometry
#define	XmCHSpace	XtCHSpace
#define	XmCHeight	XtCHeight
#define	XmCHeightInc	XtCHeightInc
#define	XmCIconMask	XtCIconMask
#define	XmCIconName	XtCIconName
#define	XmCIconNameEncoding	XtCIconNameEncoding
#define	XmCIconPixmap	XtCIconPixmap
#define	XmCIconWindow	XtCIconWindow
#define	XmCIconX	XtCIconX
#define	XmCIconY	XtCIconY
#define	XmCIconic	XtCIconic
#define	XmCIndex	XtCIndex
"	XmCInitialResourcesPersistent	XtCInitialResourcesPersistent
#define	XmCInitialState	XtCInitialState
	XmCInput	XtCInput
"	XmCInsertPosition	XtCInsertPosition
	XmCInterval	XtCInterval
1	XmCJustify	XtCJustify
1	XmCLabel	XtCLabel
1	XmCLength	XtCLength
	XmCMappedWhenManaged	XtCMappedWhenManaged
1	XmCMargin	XtCMargin
1	XmCMaxAspectX	XtCMaxAspectX
1	XmCMaxAspectY	XtCMaxAspectY
#define	XmCMaxHeight	XtCMaxHeight

6-261

Figure 6-315: <Xm/XmStrDefs.h>*, Part 28 of 34

#define	XmCMaxWidth	XtCMaxWidth
#define	XmCMenuEntry	XtCMenuEntry
#define	XmCMinAspectX	XtCMinAspectX
#define	XmCMinAspectY	XtCMinAspectY
#define	XmCMinHeight	XtCMinHeight
#define	XmCMinWidth	XtCMinWidth
#define	XmCNotify	XtCNotify
#define	XmCOrientation	XtCOrientation
#define	XmCOverrideRedirect	XtCOverrideRedirect
#define	XmCParameter	XtCParameter
#define	XmCPixmap	XtCPixmap
#define	XmCPosition	XtCPosition
#define	XmCReadOnly	XtCReadOnly
#define	XmCResize	XtCResize
#define	XmCReverseVideo	XtCReverseVideo
#define	XmCSaveUnder	XtCSaveUnder
#define	XmCScreen	XtCScreen
#define	XmCScrollDCursor	XtCScrollDCursor
	XmCScrollHCursor	XtCScrollHCursor
#define	XmCScrollLCursor	XtCScrollLCursor
11	XmCScrollProc	XtCScrollProc
#define	XmCScrollRCursor	XtCScrollRCursor
#define	XmCScrollUCursor	XtCScrollUCursor
#define	XmCScrollVCursor	XtCScrollVCursor
#define	XmCSelection	XtCSelection
	XmCSelectionArray	XtCSelectionArray
	XmCSensitive	XtCSensitive
	XmCSpace	XtCSpace
	XmCString	XtCString
	XmCTextOptions	XtCTextOptions
1	XmCTextPosition	XtCTextPosition
	XmCTextSink	XtCTextSink
	XmCTextSource	XtCTextSource
	XmCThickness	XtCThickness
#define	XmCThumb	XtCThumb

6-262 LIBRARIES

Figure 6-316: <Xm/XmStrDefs.h>*, Part 29 of 34

_			
	#define	XmCTitle	XtCTitle
	#define	XmCTitleEncoding	XtCTitleEncoding
	#define	XmCTransient	XtCTransient
	#define	XmCTransientFor	XtCTransientFor
	#define	XmCTranslations	XtCTranslations
	#define	XmCVSpace	XtCVSpace
	#define	XmCValue	XtCValue
	#define	XmCVisual	XtCVisual
	#define	XmCWaitForWm	XtCWaitForWm
	#define	XmCWidget	XtRWidget
	#define	XmCWidth	XtCWidth
	#define	XmCWidthInc	XtCWidthInc
	#define	XmCWinGravity	XtCWinGravity
	#define	XmCWindow	XtCWindow
	#define	XmCWindowGroup	XtCWindowGroup
	#define	XmCWmTimeout	XtCWmTimeout
	#define	XmCX	XtCX
	#define	XmCY	XtCY
	#define	XmNaccelerators	XtNaccelerators
	#define	XmNallowShellResize	XtNallowShellResize
	#define	XmNancestorSensitive	XtNancestorSensitive
	#define	XmNargc	XtNargc
	#define	XmNargv	XtNargv
	#define	XmNbackground	XtNbackground
	#define	XmNbackgroundPixmap	XtNbackgroundPixmap
	#define	XmNbaseHeight	XtNbaseHeight
	#define	XmNbaseHeight	XtNbaseHeight
		XmNbaseWidth	XtNbaseWidth
	**	XmNbaseWidth	XtNbaseWidth
		XmNbitmap	XtNbitmap
	**	XmNborder	XtNborder
	**	XmNborderColor	XtNborderColor
		XmNborderPixmap	XtNborderPixmap
		XmNborderWidth	XtNborderWidth
	#define	XmNcallback	XtNcallback

Figure 6-317: <Xm/XmStrDefs.h>*, Part 30 of 34

#define	XmNchildren	XtNchildren
#define	XmNcolormap	XtNcolormap
#define	XmNcreatePopupChildProc	XtNcreatePopupChildProc
#define	XmNdepth	XtNdepth
#define	XmNdestroyCallback	XtNdestroyCallback
#define	XmNeditType	XtNeditType
#define	XmNfile	XtNfile
#define	XmNfont	XtNfont
#define	XmNfontSet	XtNfontSet
#define	XmNforceBars	XtNforceBars
#define	XmNforeground	XtNforeground
#define	XmNfunction	XtNfunction
#define	XmNgeometry	XtNgeometry
#define	XmNheight	XtNheight
#define	XmNheightInc	XtNheightInc
#define	XmNhighlight	XtNhighlight
#define	XmNiconMask	XtNiconMask
#define	XmNiconName	XtNiconName
#define	XmNiconNameEncoding	XtNiconNameEncoding
#define	XmNiconPixmap	XtNiconPixmap
#define	XmNiconWindow	XtNiconWindow
#define	XmNiconX	XtNiconX
#define	XmNiconY	XtNiconY
#define	XmNiconic	XtNiconic
#define	XmNindex	XtNindex
#define	XmNinitialResourcesPersistent	XtNinitialResourcesPersistent
#define	XmNinitialState	XtNinitialState
#define	XmNinnerHeight	XtNinnerHeight
#define	XmNinnerWidth	XtNinnerWidth
#define	XmNinnerWindow	XtNinnerWindow
#define	XmNinput	XtNinput
#define	XmNinsertPosition	XtNinsertPosition
	XmNinternalHeight	XtNinternalHeight
	XmNinternalWidth	XtNinternalWidth
#define	XmNjumpProc	XtNjumpProc

6-264 LIBRARIES

Figure 6-318: <Xm/XmStrDefs.h>*, Part 31 of 34

#define	XmNjustify	XtNjustify
#define	XmNlength	XtNlength
#define	XmNlowerRight	XtNlowerRight
#define	XmNmappedWhenManaged	XtNmappedWhenManaged
#define	XmNmaxAspectX	XtNmaxAspectX
#define	XmNmaxAspectY	XtNmaxAspectY
#define	XmNmaxHeight	XtNmaxHeight
#define	XmNmaxWidth	XtNmaxWidth
#define	XmNmenuEntry	XtNmenuEntry
#define	XmNminAspectX	XtNminAspectX
#define	XmNminAspectY	XtNminAspectY
#define	XmNminHeight	XtNminHeight
#define	XmNminWidth	XtNminWidth
#define	XmNname	XtNname
#define	XmNnotify	XtNnotify
#define	XmNnumChildren	XtNnumChildren
#define	XmNorientation	XtNorientation
#define	XmNoverrideRedirect	XtNoverrideRedirect
#define	XmNparameter	XtNparameter
#define	XmNpixmap	XtNpixmap
#define	XmNpopdownCallback	XtNpopdownCallback
#define	XmNpopupCallback	XtNpopupCallback
#define	XmNresize	XtNresize
#define	XmNreverseVideo	XtNreverseVideo
#define	XmNsaveUnder	XtNsaveUnder
#define	XmNscreen	XtNscreen
#define	XmNscrollDCursor	XtNscrollDCursor
#define	XmNscrollHCursor	XtNscrollHCursor
#define	XmNscrollLCursor	XtNscrollLCursor
#define	XmNscrollProc	XtNscrollProc
	XmNscrollRCursor	XtNscrollRCursor
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	XmNscrollUCursor	XtNscrollUCursor
	XmNscrollVCursor	XtNscrollVCursor
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	XmNselection	XtNselection
#define	XmNselectionArray	XtNselectionArray

Figure 6-319: <Xm/XmStrDefs.h>*, Part 32 of 34

#define	XmNsensitive	XtNsensitive
#define	XmNshown	XtNshown
#define	XmNspace	XtNspace
#define	XmNstring	XtNstring
#define	XmNtextOptions	XtNtextOptions
#define	XmNtextSink	XtNtextSink
#define	XmNtextSource	XtNtextSource
#define	XmNthickness	XtNthickness
#define	XmNthumb	XtNthumb
#define	XmNthumbProc	XtNthumbProc
#define	XmNtitle	XtNtitle
#define	XmNtitleEncoding	XtNtitleEncoding
#define	XmNtop	XtNtop
#define	XmNtransient	XtNtransient
#define	XmNtransientFor	XtNtransientFor
#define	XmNtransientFor	XtNtransientFor
#define	XmNtranslations	XtNtranslations
#define	XmNupdate	XtNupdate
#define	XmNuseBottom	XtNuseBottom
#define	XmNuseRight	XtNuseRight
#define	XmNvalue	XtNvalue
#define	XmNvisual	XtNvisual
#define	XmNwaitForWm	XtNwaitForWm
#define	XmNwidth	XtNwidth
#define	XmNwidthInc	XtNwidthInc
#define	XmNwinGravity	XtNwinGravity
#define	XmNwindow	XtNwindow
#define	XmNwindowGroup	XtNwindowGroup
''	XmNwmTimeout	XtNwmTimeout
#define		XtNx
#define	-	XtNy
#define	XmRAcceleratorTable	XtRAcceleratorTable
#define	XmRAtom	XtRAtom
	XmRBitmap	XtRBitmap
#define	XmRBool	XtRBool

6-266 LIBRARIES

Figure 6-320: <Xm/XmStrDefs.h>*, Part 33 of 34

#define	XmRBoolean	XtRBoolean
#define	XmRCallProc	XtRCallProc
#define	XmRCallback	XtRCallback
#define	XmRCardinal	XtRCardinal
#define	XmRColor	XtRColor
#define	XmRColormap	XtRColormap
#define	XmRCursor	XtRCursor
#define	XmRDimension	XtRDimension
#define	XmRDisplay	XtRDisplay
#define	XmREditMode	XtREditMode
#define	XmREnum	XtREnum
#define	XmRFile	XtRFile
#define	XmRFloat	XtRFloat
#define	XmRFont	XtRFont
#define	XmRFontSet	XtRFontSet
#define	XmRFontStruct	XtRFontStruct
#define	XmRFunction	XtRFunction
#define	XmRGeometry	XtRGeometry
#define	XmRImmediate	XtRImmediate
#define	XmRInitialState	XtRInitialState
#define		XtRInt
	XmRJustify	XtRJustify
	XmRLongBoolean	XtRLongBoolean
,, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	XmROrientation	XtROrientation
	XmRObject	XtRObject
,, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	XmRPixel	XtRPixel
	XmRPixmap	XtRPixmap
	XmRPointer	XtRPointer
,, ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	XmRPosition	XtRPosition
1	XmRScreen	XtRScreen
1	XmRShort	XtRShort
	XmRString	XtRString
	XmRStringArray	XtRStringArray
	XmRStringTable	XtRStringTable
#define	XmRTextPosition	XtCTextPosition

Figure 6-321: <Xm/XmStrDefs.h>*, Part 34 of 34

#define XmRTranslationTable	XtRTranslationTable
#define XmRUnsignedChar	XtRUnsignedChar
#define XmRVisual	XtRVisual
#define XmRWidget	XtRWidget
#define XmRWidgetClass	XtRWidgetClass
#define XmRWidgetList	XtRWidgetList
#define XmRWindow	XtRWindow

6-268 LIBRARIES

TCP/IP Data Definitions



This section is new to the Third Edition of this document, but will not be marked with the "G" diff-mark.

This section contains standard data definitions that describe system data for the optional TCP/IP Interfaces. These data definitions are referred to by their names in angle brackets: <name.h> and <sys/name.h>. Included in these data definitions are macro definitions and structure definitions. While an ABI-conforming system may provide TCP/IP interfaces, it need not contain the actual data definitions referenced here. Programmers should observe that the sources of the structures defined in these data definitions are defined in SVID.

System Data Interfaces

6-269

```
Figure 6-322: <netinet/in.h>
```

```
#define
             IPPROTO IP
#define
             IPPROTO_TCP 6
struct in_addr {
      union {
             struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
             struct { u_short s_w1,s_w2; } S_un_w;
             u_long S_addr;
      } S_un;
#define
             s_addr
                          S_un.S_addr
};
#define
             INADDR_ANY
                                       (u_long)0x00000000
#define
             INADDR_LOOPBACK
                                       (u_long)0x7F000001
#define
             INADDR_BROADCAST
                                       (u_long)0xffffffff
             IN_SET_LOOPBACK_ADDR(a) \
#define
      {(a)->sin_addr.s_addr = htonl(INADDR_LOOPBACK); \
      (a)->sin_family = AF_INET;}
struct sockaddr_in {
      short
                   sin_family;
                   sin_port;
      u short
      struct in_addr
                         sin_addr;
                   sin zero[8];
};
             IP_OPTIONS
#define
                          1
```

M

6-270 LIBRARIES

Figure 6-323: <netinet/ip.h>

#define	IPOPT_EOL	0
#define	IPOPT_NOP	1
#define	IPOPT_LSRR	131
#define	IPOPT_SSRR	137

Figure 6-324: <netinet/tcp.h>

#define	TCP_NODELAY	0x01

System Data Interfaces

6-271

7 DEVELOPMENT ENVIRONMENT

Development Commands	7-1
PATH Access to Development Tools	7-1
Software Packaging Tools	7.0
	7-2
System Headers	7-2
Static Archives	7-2

Table of Contents

Development Commands

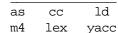


THE FACILITIES AND INTERFACES DESCRIBED IN THIS SECTION ARE OPTIONAL COMPONENTS OF THE $System\ V\ Application\ Binary\ Interface.$



This section is new to the Third Edition of this document, but will not be marked with the "G" diff-mark.

The Development Environment for Intel386 implementations of UnixWare® 2.0 will contain all of the development commands required by the System V ABI, namely;



Each command accepts all of the options required by the System V ABI, as defined in the SD_CMD section of the *System V Interface Definition, Edition 4*.

PATH Access to Development Tools

The development environment for the Intel386 System V implementations is accessible using the system default value for PATH. The default if no options are given to the cc command is to use the libraries and object file formats that are required for ABI compliance.

Development Commands

7-1

M

Software Packaging Tools

The development environment for i386 implementations of the System V ABI shall include each of the following commands as defined in the AS_CMD section of - System V Interface Definition, Edition 4.

pkgproto pkgtrans pkgmk

System Headers

Systems that do not have an ABI Development Environment may not have system header files. If an ABI Development Environment is supported, system header files will be included with the Development Environment. The primary source for contents of header files is always the *System V Interface Definition, Edition 4*. In those cases where SVID Fourth Edition doesn't specify the contents of system headers, Chapter 6 "Data Definitions" of this document shall define the associations of data elements to system headers for compilation. For greatest source portability, applications should only depend on header file contents defined in SVID.

Static Archives

Level 1 interfaces defined in *System V Interface Definition, Edition 4*, for each of the following libraries, may be statically linked safely into applications. The resulting executable will not be made non-compliant to the ABI solely because of the static linkage of such members in the executable.

libcurses libm

The archive libcurses.a is located in /usr/lib on conforming i386 development M environments. The archive libm.a is located in /usr/lib on conforming i386 development environments.

7-2

DEVELOPMENT ENVIRONMENT

8 EXECUTION ENVIRONMENT

Application Environment	8-1
The /dev Subtree	8-1

Table of Contents i

Application Environment



This section is new to the Third Edition of this document, but will not be marked with the "G" diff-mark.

This section specifies the execution environment information available to application programs running on an i386 ABI-conforming computer.

The /dev Subtree

All networking device files described in the Generic ABI shall be supported on all i386 ABI-conforming computers. In addition, the following device files are required to be present on all i386 ABI-conforming computers.

/dev/null This device file is a special "null" device that may be

used to test programs or provide a data sink. This file is

writable by all processes.

/dev/tty This device file is a special one that directs all output to

the controlling TTY of the current process group. This

file is readable and writable by all processes.

/dev/sxtXX

/dev/ttyXX These device files, where XX represents a two-digit

integer, represent device entries for terminal sessions.

All these device files must be examined by the

ttyname() call. Applications must not have the device names of individual terminals hard-coded within them. The sxt entries are optional in the system but, if present

must be included in the library routine's search.

The following device files are required to be present on all i386 ABI-conforming computers that support the corresponding hardware devices.

Application Environment

8-1

/dev/lpX

This device file is the lineprinter device. The letter "X" represents a one-digit integer that identifies the particular lineprinter device.

/dev/dsk/ /dev/rdsk/

These directories contain the raw and block disk device files. They are of the form:

f[01][t] f[01][35][dh][t] c#t#d#s#

where 'c' is followed by a controller number,

't' is followed by a target number, 'd' is followed by a disk unit number, 's' is followed by a disk slice number.

/dev/rmt/

These directories contain the raw and block tape device files. The devices guaranteed to be in this directory are:

ctape1 ntape1

/dev/cdrom/ /dev/rcdrom/

These directories contain the raw and block CD-ROM disk device files. They are of the form:

c#t#1# c#t#1# cdrom#

The letter 'c' is followed by a controller number. The letter 't' is followed by a target number on the controller. The letter 'l' is followed by a logical unit number on the target. The device "cdrom" is followed by a sequential number as nodes are created.

No leading zeroes are used in the numbers (target four is t4 not t04). The numbering for 'c', 't' and 'l' begins at zero and the numbering for 'n' begins at one.

8-2

EXECUTION ENVIRONMENT



Table of Contents i

Index

ABI Compliance 7: 1 ABI conformance 3: 1, 3, 5, 18, 24, 28–29, 34 application 6: 4 see also undefined behavior 3: 1 see also unspecified property 3: 1 system 6: 4 ABI Development Environment 7: 2	implementation 3: 1 processor 3: 1 archive file 7: 2 argc 3: 26 arguments bad assumptions 3: 44 exec(BA_OS) 3: 26 floating-point 3: 17 function 3: 9
absolute code 3: 34, 5: 3 see also position-independent code 3: 34	integer 3: 17 main 3: 26 order 3: 11 pointer 3: 17
address absolute 3: 46 stack 3: 29 stack object 3: 46 unaligned 3: 3 virtual 5: 1 addressing, virtual (see virtual addressing) aggregate 3: 3 aio.h 6: 6 alignment array 3: 3 bit-field 3: 6 double 3: 2-3, 5, 18 executable file 5: 1	sign extension 3: 17 stack 3: 11, 17 structure and union 3: 18 variable list 3: 44 argv 3: 26 array 3: 3 ArrowBG.h 6: 193 ArrowB.h 6: 193 as 7: 1 assert.h 6: 6 Atom.h 6: 130 automatic variables 3: 43 auxiliary vector 3: 29
scalar types 3: 2 stack 3: 11	В
stack frame 3: 10, 44 structure and union 3: 3	base address 3: 31, 4: 4, 5: 3 behavior, undefined (see undefined behavior)
allocation, dynamic stack space 3: 43–44 ANSI, C (see C language, ANSI) ANSI C 3: 28 Application Environment 8: 1 architecture	behavior) bit-field 3: 6 alignment 3: 6 allocation 3: 6 unnamed 3: 6 bounds check fault 3: 25

Index IN-1

branch instructions 3: 41 breakpoint trap 3: 25 BulletinB.h 6: 193	uninitialized 5: 2 data representation 3: 1 Development Environment 7: 1–2 DialogS.h 6: 195
С	dirent.h 6:8 Display.h 6:196 divide error fault 3:25
C language ANSI 3: 2, 26, 44, 6: 5 calling sequence 3: 9, 44 fundamental types 3: 2 main 3: 26 portability 3: 44 switch statements 3: 41 call by value 3: 18 call instruction 3: 13, 15, 36, 39–40	dlfcn.h 6:8 double 3:2 double fault abort 3:25 doubleword 3:1,3 DragC.h 6:201 DragIcon.h 6:201 DragOverS.h 6:201 DrawingA.h 6:202 DrawnB.h 6:202
calling sequence 3: 9 function epilogue 3: 13, 15–16, 37 function prologue 3: 13, 15, 37 function prologue and epilogue 3: 36	DropSMgr.h 6: 204 DropTrans.h 6: 205 dynamic frame size 3: 43 dynamic linking 3: 14, 17, 20, 5: 5 environment 5: 10
CascadeBG.h 6: 194 Cascade.h 6: 193 char 3: 2 code generation 3: 34 code sequences 3: 34 Command.h 6: 194 Composite.h 6: 131 configuration parameters (see tunable	lazy binding 5: 10 LD_BIND_NOW 5: 10 relocation 5: 5, 9 see also dynamic linker 5: 5 dynamic segments 3: 21, 5: 3 dynamic stack allocation 3: 44 signals 3: 45
parameters) Constraint.h 6: 131 coprocessor error fault 3: 25	E
coprocessor error radit 3.25 coprocessor overrun abort 3:25 Core.h 6:131 ctype.h 6:7 cursorfont.h 6:134 CutPaste.h 6:195	EFLAGS register, initial value 3: 26 EIP-relative 3: 35, 41 ELF 4: 1 elf.h 6: 14 emulation, instructions 3: 1 environment 5: 10
D	exec(BA_OS) 3: 26 envp 3: 26
data process 3: 20	errno.h 6:17 exceptions 3:24

IN-2 Index

interface 3: 24	initial value 3: 29
signals 3: 24	frame size, dynamic 3: 44
exec(BA OS) 3: 28, 35	Frame.h 6:206
interpreter 3: 31	ftw.h 6:23
paging 5: 1	function, void 3: 12
process initialization 3: 26	function arguments (see arguments)
executable file, segments 5: 3	function call, code 3: 39
execution mode (see processor execu-	function linkage (see calling
tion mode)	sequence)
_exit(BA OS) 3:24	function prologue and epilogue (see
extended instruction pointer, relative	calling sequence)
addressing (see EIP-relative)	
extended-precision 3:13	_
•	G
_	general protection fault /about 0.05
F	general protection fault/abort 3: 25
fcntl.h 6:19	global offset table 3: 14, 17, 35, 4: 2, 4–6,
	5: 5
file, object (see object file) file offset 5: 1	ebx 3:36
FileSB.h 6: 205	_GLOBAL_OFFSET_TABLE_ 3: 36 relocation 3: 35
float 3:2	_GLOBAL_OFFSET_TABLE_ (see global offset table)
float.h 6:21	glob.h 6: 24
floating-point	3
arguments 3: 17	grp.h 6:25
return value 3: 12	
floating-point control register, initial value 3: 26	Н
	11
floating-point status register, initial value 3: 26	halfword 3: 1
	heap, dynamic stack 3: 44
fmtmsg.h 6:22	
fnmatch.h 6: 22 formats	
	1
array 3: 3	iconv.h 6:25
structure 3: 3	
union 3:3	
Form.h 6: 205	in.h 6:270
FORTRAN	initialization, process 3: 26
COMMON 3: 3	instructions, emulation 3: 1
EQUIVALENCE 3:3	int 3:2
_fpstart 3: 28	int instruction 3: 25
frame pointer 3: 11, 37	integer arguments 3: 17

M Intel386 3: 1, 9, 24, 44, 5: 1 Intel387 3: 1 m4 7:1 interrupt 3: 12 Machine Status Word register, initial Intrinsic.h 6:140 value 3: 26 invalid opcode fault 3: 25 machlock.h 6:32 invalid TSS fault 3: 25 main ip.h 6:271 arguments 3: 26 declaration 3: 26 MainW.h 6:207 malloc(BA OS) 3: 22-23 jmp instruction 5:8 math.h 6:32 memory allocation, stack 3: 43-44 memory management 3: 20 ı MenuShell.h 6:207 MessageB.h 6:208 LabelG.h 6:206 mmap(KE_OS) 3: 23 Label.h 6:206 MrmPublic.h 6:212 langinfo.h 6:28 MwmUtil.h 6:214 lazy binding 5: 10 lcall instruction 3:24 ld 7:1 Ν LD_BIND_NOW 5:10 ld(SD CMD) (see link editor) netconfig.h 6:38 lex 7:1 netdir.h 6:40 libc 6:2 nl_types.h 6:40 libcurses 7: 2 no coprocessor fault 3: 25 libm 7: 2 nonmaskable interrupt 3: 25 limits.h 6:30 null pointer 3: 3, 21, 26 link editor 4: 6, 5: 5 linkage, function (see calling O sequence) List.h 6:207 object file 4: 1 local variables 3:43 ELF header 4: 1 locale.h 6:31 executable 3: 35 long 3:2 executable file 3: 35 long double 3:2 relocation 4: 4 longjmp(BA LIB) (see section 4: 2 setjmp(BA_LIB)) see also archive file 4:1 lwpsynch.h 6:32 see also dynamic linking 5: 5 see also executable file 4: 1 see also relocatable file 4: 1

IN-4

see also shared object file 4: 1 segment 5: 1	see also procedure linkage table 3: 35
shared object file 3: 35	priocntl.h 6:43
special sections 4: 2	procedure linkage table 3: 35, 4: 2, 5–6,
Object.h 6: 140	5: 5, 7
offset table, global (see global offset	procedures (see functions)
table)	process
optimization 3: 14, 17	dead 3: 45
overflow trap 3: 25	entry point 3: 26
overnow trap 5. 25	initialization 3: 26
	segment 3: 20
Р	size 3: 20
•	stack 3: 28
padding	virtual addressing 3: 20
arguments 3: 11, 18	processor architecture 3: 1
structure and union 3:3	processor execution mode 3: 24, 26
page fault 3: 25	processor-specific information 3: 1, 9,
page size 3: 20, 31, 5: 1	20, 34, 5: 1, 5–7, 6: 2–3, 7: 1
paging 3: 20, 5: 1	program loading 3: 31, 5: 1
performance 5: 1	PushBG.h 6: 215
PanedW.h 6: 214	PushB.h 6: 215
parameters	pushl instruction 5:8
function (see arguments)	pwd.h 6:45
system configuration (see tunable	pwa.11 0.40
parameters)	
PATH variable 7: 1	R
performance, paging 5: 1	-
physical addressing 3: 20	RectObj.h 6:140
pkgmk 7:2	re-entrancy 3: 14
pkgproto 7:2	regex.h 6:47
pkgtrans 7:2	registers
pointer 3: 3	calling sequence 3: 11
function argument 3: 17	cs, ds, es, and ss 3: 29
null 3: 3, 26	description 3: 9, 11
poll.h 6: 42	eax 3: 11
portability	ebp 3:11
C program 3: 44	ebx 3:11
instructions 3: 1	ecx 3: 12
position-independent code 3: 14, 17,	edi 3:11
34, 36, 5: 3	edx 3: 12
see also absolute code 3: 34	EFLAGS 3: 12
see also global offset table 3: 35	esi 3:11

esp 3: 11	process 3: 20–21, 5: 1, 6
floating-point 3: 9, 12	segment not present fault 3: 25
floating-point control word 3: 12	segment permissions 3: 23
floating-point return value 3: 13	segment registers 3: 29
global 3: 9	SelectioB.h 6:217
initial values 3: 29	SeparatoG.h 6:217
scratch 3: 12	Separator.h 6:217
variable 3: 11	setjmp(BA_LIB) 3:45
registers cr0 (see Machine Status	setjmp.h 6:66
Word register)	setrlimit(BA_OS) 3:23
registers eax, integer return value	shared object file 3: 35
3: 12	segments 3: 21, 5: 3
relocation	Shell.h 6:141
global offset table 3: 35	short 3:2
see object file 4: 4	sign extension
RepType.h 6:215	arguments 3: 17
resources, shared 3: 20	bit-field 3: 6
ret instruction 3: 13, 15	$signal(BA_OS)$ 3: 12, 24
return address 3: 13	signal.h 6:69
return value	signals 3: 12, 45
floating-point 3: 12	signed 3:2,6
integer 3: 12	single step trap/fault 3: 25
pointer 3: 12	sizeof 3:2
structure and union 3: 14	structure 3: 3
RowColumn.h 6:216	Ssape.h 6:141
rpc.h 6:63	stack
rtpriocntl.h 6:63	address 3: 29
	dynamic allocation 3: 44
	growth 3: 10
S	initial process 3: 28
scalar types 2: 2	process 3: 20–21
scalar types 3: 2 Scale.h 6: 216	system management 3: 22
Screen.h 6:216	stack exception fault 3: 25
ScrollBar.h 6:216	stack frame 3: 9–10, 13–15, 17, 37, 43
ScrolledW.h 6:217	alignment 3: 10, 29, 44
search.h 6:64	organization 3: 10, 43
secondary storage 3: 20	size 3: 11, 43
section, object file 5: 1	stack pointer 3: 11, 29, 37
segment	initial value 3: 29
dynamic 3: 21	<stdarg.h> 3:44</stdarg.h>
permissions 5: 2	stdarg.h 6:76
permissions 5.2	stddef.h 6:76

IN-6 Index

stdio.h 6:79	text
stdlib.h 6:79	process 3: 20
stropts.h 6:85	sharing 3: 35
structure 3: 3	TextF.h 6: 218
function argument 3: 18	Text.h 6: 218
padding 3: 3	<thread> 6:100</thread>
return value 3: 14	ticlts.h 6:100
supervisor mode (see processor execu-	ticots.h 6:101
tion mode)	ticotsord.h 6:101
SVID 7: 2	time.h 6:102
switch statements 3:41	tiuser.h 6:112
<synch> 6:88</synch>	ToggleBG.h 6:218
sysconf(BA_OS) 3: 20, 31	ToggleB.h 6:218
sys/ipc.h 6:26	tspriocntl.h 6:113
sys/mman.h 6:33	tunable parameters, stack size 3: 23
sys/mod.h 6:34	type mismatch 3: 15
sys/mount.h 6:35	
sys/msg.h 6:36	
sys/param.h 6:41	U
sys/procset.h 6:44	22 22 22 22 22 22 22 22 22 22 22 22 22
sys/resource.h 6:48	ucontext.h 6:116
sys/sem.h 6:65	ulimit.h 6: 117
sys/shm.h 6:66	unaligned address (see address,
sys/siginfo.h 6:72	unaligned)
sys/stat.h 6:74	undefined behavior 3: 1, 12, 15, 28–29
sys/statvfs.h 6:75	5: 2
sys/sysi86.h 6:88	see also ABI conformance 3: 1
System Headers 7: 2	see also unspecified property 3: 1
system load 3: 20	uninitialized data 5:2
System V ABI 6: 4	union 3: 3
sys/time.h 6:103	function argument 3: 18
sys/times.h 6:104	return value 3: 14
sys/types.h 6:114	unistd.h 6:119
sys/uio.h 6:117	unsigned 3:2,6
sys/utsname.h 6:120	unspecified property 3: 1, 12, 14, 24,
•	26–27, 29, 5: 1–2
_	see also ABI conformance 3: 1
T	see also undefined behavior 3: 1
	user mode (see processor execution
tcpip/tcp.h 6:271	mode)
termination, process 3: 45	utime.h 6:119
termios.h 6:98	

V

<varargs.h> 3: 44
variable argument list 3: 17, 44
variables, automatic 3: 43
Vendor.h 6: 141
VendorS.h 6: 219
VirtKeys.h 6: 220
virtual addressing 3: 20, 35
bounds 3: 21
void functions 3: 12

W

wait.h 6: 121
wchar.h 6: 122
wctype.h 6: 125
word 3: 1, 10
wordexp.h 6: 126

X

Xcms.h 6: 158
X.h 6: 153
Xlib.h 6: 184
Xm.h 6: 234
XmStrDefs.h 6: 268
Xresource.h 6: 186
Xutil.h 6: 191

Y

yacc 7:1

Z

zero null pointer 3: 3 uninitialized data 5: 2

IN-8 Index