SYSTEM V
APPLICATION
BINARY INTERFACE

Intel386™ Architecture
Processor Supplement

Fourth Edition
# Contents

## Table of Contents

- INTRODUCTION
- SOFTWARE INSTALLATION
- LOW-LEVEL SYSTEM INFORMATION
- OBJECT FILES
- PROGRAM LOADING AND DYNAMIC LINKING
- LIBRARIES
- DEVELOPMENT ENVIRONMENT
- EXECUTION ENVIRONMENT
- Index

## 1. INTRODUCTION

- The Intel386 Architecture and the System V ABI
  - 1-1
- How to Use the Intel386 Architecture ABI Supplement
  - 1-2

## 2. SOFTWARE INSTALLATION

- Software Distribution Formats
  - 2-1

## 3. LOW-LEVEL SYSTEM INFORMATION

- Machine Interface
  - 3-1
- Function Calling Sequence
  - 3-9
- Operating System Interface
  - 3-20
- Coding Examples
  - 3-34
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
iv Table of Contents
Figure 6-21: <float.h>, Single-Precision 6-20
Figure 6-22: <float.h>, Double-Precision 6-20
Figure 6-23: <float.h>, Extended-Precision 6-21
Figure 6-24: <fmtmsg.h>, Part 1 of 2 6-21
Figure 6-25: <fmtmsg.h>, Part 2 of 2 6-22
Figure 6-26: <fnmatch.h> 6-22
Figure 6-27: <ftw.h> 6-23
Figure 6-28: <glob.h> 6-24
Figure 6-29: <grp.h> 6-25
Figure 6-30: <iconv.h> 6-25
Figure 6-31: <sys/ipc.h> 6-26
Figure 6-32: <langinfo.h>, Part 1 of 2 6-27
Figure 6-33: <langinfo.h>, Part 2 of 2 6-28
Figure 6-34: <limits.h>, Part 1 of 2 6-29
Figure 6-35: <limits.h>, Part 2 of 2 6-30
Figure 6-36: <locale.h> 6-31
Figure 6-37: <lwpysynch.h> 6-32
Figure 6-38: <machlock.h> 6-32
Figure 6-39: <math.h> 6-32
Figure 6-40: <sys/mman.h> 6-33
Figure 6-41: <sys/mod.h> 6-34
Figure 6-42: <sys/mount.h> 6-35
Figure 6-43: <sys/msg.h> 6-36
Figure 6-44: <netconfig.h>, Part 1 of 2 6-37
Figure 6-45: <netconfig.h>, Part 2 of 2 6-38
Figure 6-46: <netdir.h>, Part 1 of 2 6-39
Figure 6-47: <netdir.h>, Part 2 of 2 6-40
Figure 6-48: <nl_types.h> 6-40
Figure 6-49: <sys/param.h> 6-41
Figure 6-50: <poll.h> 6-42
Figure 6-51: <sys/priocnt1.h> 6-43
Figure 6-52: <sys/procset.h> 6-44
Figure 6-53: <pwd.h> 6-45
Figure 6-54: <regex.h>*, Part 1 of 2 6-46
Figure 6-55: <regex.h>*, Part 2 of 2 6-47
Figure 6-56: <sys/resource.h> 6-48
Figure 6-57: <rpc.h>, Part 1 of 16 6-49
Figure 6-58: <rpc.h>, Part 2 of 16 6-50
Figure 6-59: <rpc.h>, Part 3 of 16 6-51
Figure 6-60: <rpc.h>, Part 4 of 16 6-52
Figure 6-61: <rpc.h>, Part 5 of 16 6-53
Figure 6-62: <rpc.h>, Part 6 of 16 6-54
Figure 6-63: <rpc.h>, Part 7 of 16 6-55
Figure 6-64: <rpc.h>, Part 8 of 16 6-56
Figure 6-65: <rpc.h>, Part 9 of 16 6-57
Figure 6-111: <termios.h>, Part 8 of 10 6-96
Figure 6-112: <termios.h>, Part 9 of 10 6-97
Figure 6-113: <termios.h>, Part 10 of 10 6-98
Figure 6-114: <thread.h>*, Part 1 of 2 6-99
Figure 6-115: <thread.h>*, Part 2 of 2 6-100
Figure 6-116: <sys/tclts.h> 6-100
Figure 6-117: <sys/ticots.h> 6-101
Figure 6-118: <sys/ticotsord.h> 6-101
Figure 6-119: <time.h>* 6-102
Figure 6-120: <sys/time.h> 6-103
Figure 6-121: <sys/times.h> 6-104
Figure 6-122: <tiuser.h>, Error Return Values 6-105
Figure 6-123: <tiuser.h>, Event Bitmasks 6-106
Figure 6-124: <tiuser.h>, Flags 6-106
Figure 6-125: <tiuser.h>, Service Types 6-107
Figure 6-126: <tiuser.h>, Values for flags field in t_info structure 6-107
Figure 6-127: <tiuser.h>, Transport Interface Data Structures, 1 of 2 6-108
Figure 6-128: <tiuser.h>, Transport Interface Data Structures, 2 of 2 6-109
Figure 6-129: <tiuser.h>, Structure Types 6-110
Figure 6-130: <tiuser.h>, Fields of Structures 6-110
Figure 6-131: <tiuser.h>, Transport Interface States 6-111
Figure 6-132: <tiuser.h>, User-level Events 6-112
Figure 6-133: <tsprocctl.h>* 6-113
Figure 6-134: <sys/types.h> 6-114
Figure 6-135: <ucontext.h>, Part 1 of 2 6-115
Figure 6-136: <ucontext.h>, Part 2 of 2 6-116
Figure 6-137: <sys/uio.h> 6-117
Figure 6-138: <ulimit.h> 6-117
Figure 6-139: <unistd.h>, Part 1 of 2 6-117
Figure 6-140: <unistd.h>, Part 2 of 2 6-119
Figure 6-141: <utime.h> 6-119
Figure 6-142: <sys/utsname.h> 6-120
Figure 6-143: <wait.h> 6-121
Figure 6-144: <wchar.h> 6-122
Figure 6-145: <wctype.h>* , Part 1 of 3 6-123
Figure 6-146: <wctype.h>* , Part 2 of 3 6-124
Figure 6-147: <wctype.h>* , Part 3 of 3 6-125
Figure 6-148: <wordexp.h>* 6-126
Figure 6-149: <X11/Atom.h>, Part 1 of 3 6-128
Figure 6-150: <X11/Atom.h>, Part 2 of 3 6-129
Figure 6-151: <X11/Atom.h>, Part 3 of 3 6-130
Figure 6-152: <X11/Composite.h> 6-131
Figure 6-153: <X11/Constraint.h> 6-131
Figure 6-154: <X11/Core.h> 6-131
Figure 6-155: <X11/curContextMenu.h>, Part 1 of 3 6-132

Table of Contents vii
Figure 6-156: `<X11/cursorfont.h>`, Part 2 of 3 6-133
Figure 6-157: `<X11/cursorfont.h>`, Part 3 of 3 6-134
Figure 6-158: `<X11/Intrinsic.h>`, Part 1 of 6 6-135
Figure 6-159: `<X11/Intrinsic.h>`, Part 2 of 6 6-136
Figure 6-160: `<X11/Intrinsic.h>`, Part 3 of 6 6-137
Figure 6-161: `<X11/Intrinsic.h>`, Part 4 of 6 6-138
Figure 6-162: `<X11/Intrinsic.h>`, Part 5 of 6 6-139
Figure 6-163: `<X11/Intrinsic.h>`, Part 6 of 6 6-140
Figure 6-164: `<X11/Object.h>` 6-140
Figure 6-165: `<X11/RectObj.h>` 6-140
Figure 6-166: `<X11/extensions/shape.h>` 6-141
Figure 6-167: `<X11/Shell.h>` 6-141
Figure 6-168: `<X11/Vendor.h>` 6-141
Figure 6-169: `<X11/X.h>`, Part 1 of 12 6-142
Figure 6-170: `<X11/X.h>`, Part 2 of 12 6-143
Figure 6-171: `<X11/X.h>`, Part 3 of 12 6-144
Figure 6-172: `<X11/X.h>`, Part 4 of 12 6-145
Figure 6-173: `<X11/X.h>`, Part 5 of 12 6-146
Figure 6-174: `<X11/X.h>`, Part 6 of 12 6-147
Figure 6-175: `<X11/X.h>`, Part 7 of 12 6-148
Figure 6-176: `<X11/X.h>`, Part 8 of 12 6-149
Figure 6-177: `<X11/X.h>`, Part 9 of 12 6-150
Figure 6-178: `<X11/X.h>`, Part 10 of 12 6-151
Figure 6-179: `<X11/X.h>`, Part 11 of 12 6-152
Figure 6-180: `<X11/X.h>`, Part 12 of 12 6-153
Figure 6-181: `<X11/Xcms.h>`, Part 1 of 5 6-154
Figure 6-182: `<X11/Xcms.h>`, Part 2 of 5 6-155
Figure 6-183: `<X11/Xcms.h>`, Part 3 of 5 6-156
Figure 6-184: `<X11/Xcms.h>`, Part 4 of 5 6-157
Figure 6-185: `<X11/Xcms.h>`, Part 5 of 5 6-158
Figure 6-186: `<X11/Xlib.h>`, Part 1 of 27 6-159
Figure 6-187: `<X11/Xlib.h>`, Part 2 of 27 6-159
Figure 6-188: `<X11/Xlib.h>`, Part 3 of 27 6-160
Figure 6-189: `<X11/Xlib.h>`, Part 4 of 27 6-161
Figure 6-190: `<X11/Xlib.h>`, Part 5 of 27 6-162
Figure 6-191: `<X11/Xlib.h>`, Part 6 of 27 6-163
Figure 6-192: `<X11/Xlib.h>`, Part 7 of 27 6-164
Figure 6-193: `<X11/Xlib.h>`, Part 8 of 27 6-165
Figure 6-194: `<X11/Xlib.h>`, Part 9 of 27 6-166
Figure 6-195: `<X11/Xlib.h>`, Part 10 of 27 6-167
Figure 6-196: `<X11/Xlib.h>`, Part 11 of 27 6-168
Figure 6-197: `<X11/Xlib.h>`, Part 12 of 27 6-169
Figure 6-198: `<X11/Xlib.h>`, Part 13 of 27 6-170
Figure 6-199: `<X11/Xlib.h>`, Part 14 of 27 6-171
Figure 6-200: `<X11/Xlib.h>`, Part 15 of 27 6-172
1 INTRODUCTION

The Intel386 Architecture and the System V ABI

How to Use the Intel386 Architecture ABI Supplement
Evolution of the ABI Specification
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.

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.
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)
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.
2 SOFTWARE INSTALLATION

Software Distribution Formats

Physical Distribution Media 2-1
File System Formats 2-1
■ s5 File System 2-1
■ UFS File System 2-3
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.
- CD-ROM optical disks.
- 150 MB quarter-inch tape.

**CAUTION**
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:
#define NICFREE 50
#define NICINO 100

struct filesys {
  u_short s_isize;
  dadtr_t s_fsize;
  short s_nfree;
  dadtr_t s_free[NICFREE];
  short s_ninode;
  ushort_t s_inode[NICINO];
  char s_flock;
  char s_ilock;
  char s_ronly;
  time_t s_time;
  short s_dinfo[4];
  dadtr_t s_tfree;
  ushort_t s_tinode;
  char s_fname[6];
  char s_fpack[6];
  long s_fill[12];
  long s_state;
  long s_magic;
  long s_type;
};

#define FsMAGIC 0xFD187E20
#define Fs1b 1
#define Fs2b 2
#define Fs4b 3
#define FsOKAY 0x7C269D38
#define FsACTIVE 0x5E72D81A
#define FsBAD 0xCB096F43
#define FsBADBLK 0XBADD8C14B

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 Fs1b, 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.
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.
struct csum {
    long    cs_ndir;
    long    cs_nbfree;
    long    cs_nifree;
    long    cs_nffree;
};

struct fs {
    struct fs *fs_link;
    struct fs *fs_rlink;
    daddr_t    fs_ablkno;
    daddr_t    fs_cblkno;
    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ング;
    long    fs_bsize;
    long    fs_fsize;
    long    fs_frag;
    longq    fs_minfree;
    long    fs_rotdelay;
    long    fs_rps;
    long    fs_bmask;
    long    fs_fmask;
    long    fs_bshift;
    long    fs_fshift;
    long    fs_maxcontig;
    long    fs_maxbpkg;
    long    fs_fragshift;
    long    fs_fabto$db;
    long    fs_absize;
    long    fs_cmask;
    long    fs_cshift;
    long    fs_nindir;
    long    fs_inopb;
    long    fs_nspf;
    long    fs_optim;
    long    fs_state;
    long    fs_sparecon[2];
    long    fs_id[2];
    daddr_t    fs_csaddr;
    long    fs_cssize;
    long    fs_cgsizer;
    long    fs_ntrak;
    long    fs_nsec;
    long    fs_spc;
    long    fs_ncyl;

(continued on next page)
long fs_cpg;
long fs_ipg;
long fs_fpg;
struct csum fs_cstotal;
char fs_fmod;
char fs_clean;
char fs_conly;
char fs_flags;
char fs_famt[MAXTLEN];
long fs_cgorotor;
struct csum *fs_csp[MAXCBUFFS];
long fs_cpc;
short fs_postbl[MAXCPG][NRPOS];
long fs_magic;
u_char fs_rotbl[1];
}

struct cg {
    struct cg *cg_link;
    struct cg *cg_rlink;
    time_t cg_time;
    long cg_cpg;
    short cg_ncyl;
    short cg_niblk;
    long cg_ndblk;
    struct csum cg_cs;
    long cgRotor;
    long cg_frotor;
    long cg_irotor;
    long cg_fsum[MAXFRAG];
    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 BBSIZE 8192
#define BBLOCK ((addr_t)(0))
#define BBLOCK ((addr_t)(BBLOCK + BBSIZE / DEV_BSIZE))
#define UFSROOTINO ((ino_t)2)
#define LOSTFOUNDINO (UFSROOTINO + 1)
#define NRPOS 8
#define MAXIPG 2048
#define MINBSIZE 4096
#define MAXCPG 32
#define MAXTLEN 512
#define MAXCBUFFS 32
#define FS_OPTTIME 0

(continued on next page)
The distribution of software in filesystem format is Level 2 as of January 1, 1993.
3 LOW-LEVEL SYSTEM INFORMATION

Machine Interface 3-1
Processor Architecture 3-1
Data Representation 3-1
  ■ Fundamental Types 3-2
  ■ Aggregates and Unions 3-3
  ■ Bit-Fields 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 3-26
  ■ Special Registers 3-26
  ■ Process Stack and Registers 3-28
<table>
<thead>
<tr>
<th>Coding Examples</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Model Overview</td>
<td>3-35</td>
</tr>
<tr>
<td>Position-Independent Function Prologue</td>
<td>3-36</td>
</tr>
<tr>
<td>Data Objects</td>
<td>3-37</td>
</tr>
<tr>
<td>Function Calls</td>
<td>3-39</td>
</tr>
<tr>
<td>Branching</td>
<td>3-41</td>
</tr>
<tr>
<td>C Stack Frame</td>
<td>3-43</td>
</tr>
<tr>
<td>Variable Argument List</td>
<td>3-44</td>
</tr>
<tr>
<td>Allocating Stack Space Dynamically</td>
<td>3-44</td>
</tr>
</tbody>
</table>
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.
### Fundamental Types

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

#### Figure 3-1: Scalar Types

<table>
<thead>
<tr>
<th>Type</th>
<th>C</th>
<th>Sizeof (bytes)</th>
<th>Alignment (bytes)</th>
<th>Intel86 Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>signed char</td>
<td>1</td>
<td>1</td>
<td>signed byte</td>
</tr>
<tr>
<td></td>
<td>unsigned char</td>
<td>1</td>
<td>1</td>
<td>unsigned byte</td>
</tr>
<tr>
<td>short</td>
<td>signed short</td>
<td>2</td>
<td>2</td>
<td>signed halfword</td>
</tr>
<tr>
<td></td>
<td>unsigned short</td>
<td>2</td>
<td>2</td>
<td>unsigned halfword</td>
</tr>
<tr>
<td>int</td>
<td>signed int</td>
<td>4</td>
<td>4</td>
<td>signed word</td>
</tr>
<tr>
<td>long</td>
<td>signed long</td>
<td>8</td>
<td>4</td>
<td>double-precision (IEEE)</td>
</tr>
<tr>
<td></td>
<td>unsigned int</td>
<td>4</td>
<td>4</td>
<td>unsigned word</td>
</tr>
<tr>
<td></td>
<td>unsigned long</td>
<td>4</td>
<td>4</td>
<td>extended-precision (IEEE)</td>
</tr>
</tbody>
</table>

#### Pointer

| Pointer   | any-type * ( ) | 4              | 4                 | unsigned word        |

#### Floating-point

|           | float         | 4              | 4                 | single-precision (IEEE) |
|           | 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.
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.

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

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

**Figure 3-2: Structure Smaller Than a Word**

```c
struct {
    char c;
};
```

Byte aligned, sizeof is 1

**Figure 3-3: No Padding**

```c
struct {
    char c;
    char d;
    short s;
    long n;
};
```

Word aligned, sizeof is 8

**Figure 3-4: Internal Padding**

```c
struct {
    char c;
    short s;
};
```

Halfword aligned, sizeof is 4

---

3-4

LOW-LEVEL SYSTEM INFORMATION
Figure 3-5: Internal and Tail Padding

```c
struct {
    char c;
    double d;
    short s;
} ;
```

Word aligned, sizeof is 16

```
+-----------------+---+---+
|     pad         | 1 | c |
|     d           |   |   |
|     d           |   |   |
|     pad         | 14| s |
```

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.

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

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

Word aligned, sizeof is 4

```
+-----------------+---+---+
|     pad         | 1 | c |
|     pad         | 2 | s |
|     j           |   |   |
```

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.

<table>
<thead>
<tr>
<th>Bit-field Type</th>
<th>Width $w$</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>signed char</td>
<td>1 to 8</td>
<td>$-2^{w-1}$ to $2^{w-1}-1$</td>
</tr>
<tr>
<td>char</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>unsigned char</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>signed short</td>
<td>1 to 16</td>
<td>$-2^{w-1}$ to $2^{w-1}-1$</td>
</tr>
<tr>
<td>short</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>unsigned short</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>signed int</td>
<td>1 to 32</td>
<td>$-2^{w-1}$ to $2^{w-1}-1$</td>
</tr>
<tr>
<td>int</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>enum</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>unsigned int</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>signed long</td>
<td>1 to 32</td>
<td>$-2^{w-1}$ to $2^{w-1}-1$</td>
</tr>
<tr>
<td>long</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
<tr>
<td>unsigned long</td>
<td></td>
<td>0 to $2^w-1$</td>
</tr>
</tbody>
</table>

“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.
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

```
0x01020304 01 3 02 2 03 1 04 0
31 24 19 14 8 7
```

Figure 3-9: Right-to-Left Allocation

```
struct {
    int  j:5;
    int  k:6;
    int  m:7;
};
```

Figure 3-10: Boundary Alignment

```
struct {
    short s:9;
    int  j:9;
    char c;
    short t:9;
    short u:9;
    char d;
};
```

Machine Interface
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.
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.

NOTE

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.

NOTE

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.
Figure 3-14: Processor Registers

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>%eax</td>
<td>Return value</td>
</tr>
<tr>
<td></td>
<td>%edx</td>
<td>Dividend register (divide operations)</td>
</tr>
<tr>
<td></td>
<td>%ecx</td>
<td>Count register (shift and string operations)</td>
</tr>
<tr>
<td></td>
<td>%ebx</td>
<td>Local register variable</td>
</tr>
<tr>
<td></td>
<td>%ebp</td>
<td>Stack frame pointer (optional) G</td>
</tr>
<tr>
<td></td>
<td>%esi</td>
<td>Local register variable</td>
</tr>
<tr>
<td></td>
<td>%edi</td>
<td>Local register variable</td>
</tr>
<tr>
<td></td>
<td>%esp</td>
<td>Stack pointer</td>
</tr>
<tr>
<td>Floating-point</td>
<td>%st (0)</td>
<td>floating-point stack top, return value</td>
</tr>
<tr>
<td></td>
<td>%st (1)</td>
<td>floating-point next to stack top</td>
</tr>
<tr>
<td></td>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td></td>
<td>%st (7)</td>
<td>floating-point stack bottom</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Position</th>
<th>Contents</th>
<th>Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>4n+8 (%ebp)</td>
<td>argument word n</td>
<td>Previous</td>
</tr>
<tr>
<td>8 (%ebp)</td>
<td>argument word 0</td>
<td></td>
</tr>
<tr>
<td>4 (%ebp)</td>
<td>return address</td>
<td></td>
</tr>
<tr>
<td>0 (%ebp)</td>
<td>previous %ebp (optional)</td>
<td>Current</td>
</tr>
<tr>
<td>−4 (%ebp)</td>
<td>unspecified</td>
<td></td>
</tr>
<tr>
<td>0 (%esp)</td>
<td>variable size</td>
<td></td>
</tr>
</tbody>
</table>

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.
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 %ebp, %ebx, %edi, %esi, and %esp “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 frame pointer 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 previous 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.

- **%eax**: 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.

- **%ebx**: As described below, this register serves as the global offset table 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 local registers have no specified role in the function calling sequence. A function must preserve their values for the caller.
Scratch registers have no specified role in the standard calling sequence. Functions do not have to preserve their values for the caller.

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 G entry to a function.

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.

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.

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
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**

<table>
<thead>
<tr>
<th>prologue:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pushl %ebp                    / save frame pointer</td>
</tr>
<tr>
<td>movl %esp, %ebp               / set new frame pointer</td>
</tr>
<tr>
<td>subl $80, %esp                / allocate stack space</td>
</tr>
<tr>
<td>pushl %edi                    / save local register</td>
</tr>
<tr>
<td>pushl %esi                    / save local register</td>
</tr>
<tr>
<td>pushl %ebx                    / save local register</td>
</tr>
</tbody>
</table>

An epilogue for the example that restores the state for the caller. This example returns the value in %edi by moving it to %eax.
Figure 3-17: Function Epilogue

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>movl %edi, %eax</td>
<td>set up return value</td>
</tr>
<tr>
<td>popl %ebx</td>
<td>restore local register</td>
</tr>
<tr>
<td>popl %esi</td>
<td>restore local register</td>
</tr>
<tr>
<td>popl %edi</td>
<td>restore local register</td>
</tr>
<tr>
<td>leave</td>
<td>restore frame pointer</td>
</tr>
<tr>
<td>ret</td>
<td>pop return address</td>
</tr>
</tbody>
</table>

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 %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.

NOTE: Structures and unions in this context have fixed sizes. The ABI does not specify how to handle variable sized objects.
A function that returns a structure or union also sets %eax 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 %eax 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**

<table>
<thead>
<tr>
<th>Position</th>
<th>After call</th>
<th>After ret</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>4n+4 (%esp)</td>
<td>argument word n</td>
<td>argument word n</td>
<td>4n-4 (%esp)</td>
</tr>
<tr>
<td>8 (%esp)</td>
<td>argument word 1</td>
<td>...</td>
<td>0 (%esp)</td>
</tr>
<tr>
<td>4 (%esp)</td>
<td>value address</td>
<td>argument word 1</td>
<td>undefined</td>
</tr>
<tr>
<td>0 (%esp)</td>
<td>return address</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

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

```
prologue:
popl %eax // pop return address
xchgl %eax, 0(%esp) // swap return address
                  // and return value address
pushl %ebp // save frame pointer
movl %esp, %ebp // set new frame pointer
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
  popl %esi // restore local register
  popl %edi // restore local register
  leave // restore frame pointer
  ret // pop return address
```
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 `%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.

**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.

<table>
<thead>
<tr>
<th>Call</th>
<th>Argument</th>
<th>Stack address</th>
</tr>
</thead>
<tbody>
<tr>
<td>g(1, 2, 3, (void *)0);</td>
<td>1</td>
<td>8(%ebp)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12(%ebp)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16(%ebp)</td>
</tr>
<tr>
<td></td>
<td>(void *)0</td>
<td>20(%ebp)</td>
</tr>
</tbody>
</table>

**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.
Figure 3-22: Floating-Point Arguments

<table>
<thead>
<tr>
<th>Call</th>
<th>Argument</th>
<th>Stack address</th>
</tr>
</thead>
<tbody>
<tr>
<td>h(1.414, 1,</td>
<td>word 0, 1.414</td>
<td>8 (%ebp)</td>
</tr>
<tr>
<td>2.998e10);</td>
<td>word 1, 1.414</td>
<td>12 (%ebp)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>16 (%ebp)</td>
</tr>
<tr>
<td></td>
<td>word 0, 2.998e10</td>
<td>20 (%ebp)</td>
</tr>
<tr>
<td></td>
<td>word 1, 2.998e10</td>
<td>24 (%ebp)</td>
</tr>
</tbody>
</table>

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.

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.
**Figure 3-23: Structure and Union Arguments**

<table>
<thead>
<tr>
<th>Call</th>
<th>Argument</th>
<th>Callee</th>
</tr>
</thead>
<tbody>
<tr>
<td>i(1, s);</td>
<td>1</td>
<td>8(%ebp)</td>
</tr>
<tr>
<td></td>
<td>word 0, s</td>
<td>12(%ebp)</td>
</tr>
<tr>
<td></td>
<td>word 1, s</td>
<td>16(%ebp)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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 process.
- 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.
**Figure 3-24: Virtual Address Configuration**

<table>
<thead>
<tr>
<th></th>
<th>Reserved</th>
<th>End of memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffffff</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0x80000000</td>
<td>Dynamic segments</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Process segments</td>
<td>Beginning of memory</td>
</tr>
</tbody>
</table>

**CAUTION**

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 0xc0000000. Individual systems may reserve less space, increasing processes’ virtual memory range.
Although applications may control their memory assignments, the typical arrangement appears below.

**Figure 3-25: Conventional Segment Arrangements**

<table>
<thead>
<tr>
<th>Address</th>
<th>Segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80000000</td>
<td>Dynamic segments</td>
</tr>
<tr>
<td>0x7fffffff</td>
<td>Data segment</td>
</tr>
<tr>
<td>0x8048000</td>
<td>Text segment</td>
</tr>
<tr>
<td>0</td>
<td>Stack segment</td>
</tr>
</tbody>
</table>

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.
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.
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).
### Figure 3-27: Hardware Exceptions and Signals

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

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.
Process Initialization

This section describes the machine state that exec(BA_OS) 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 main, conventionally declared in the following way.

Figure 3-28: Declaration for main

```c
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.
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 3-29: EFLAGS Register Fields

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

---

### Figure 3-30: Floating-Point Control Word

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

---

Operating System Interface 3-27
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.

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

<table>
<thead>
<tr>
<th></th>
<th>High addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unspecified</td>
<td></td>
</tr>
<tr>
<td>Information block,</td>
<td></td>
</tr>
<tr>
<td>including argument</td>
<td></td>
</tr>
<tr>
<td>strings, environment</td>
<td></td>
</tr>
<tr>
<td>strings, auxiliary</td>
<td></td>
</tr>
<tr>
<td>information</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>(size varies)</td>
<td></td>
</tr>
<tr>
<td>Unspecified</td>
<td></td>
</tr>
<tr>
<td>Null auxiliary vector</td>
<td></td>
</tr>
<tr>
<td>Auxiliary vector</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>(2-word entries)</td>
<td></td>
</tr>
<tr>
<td>0 word</td>
<td></td>
</tr>
<tr>
<td>Environment pointers</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>(one word each)</td>
<td></td>
</tr>
<tr>
<td>0 word</td>
<td></td>
</tr>
<tr>
<td>Argument pointers</td>
<td></td>
</tr>
<tr>
<td>. . .</td>
<td></td>
</tr>
<tr>
<td>(Argument count</td>
<td></td>
</tr>
<tr>
<td>words)</td>
<td></td>
</tr>
<tr>
<td>4 (%esp)</td>
<td>Low addresses</td>
</tr>
<tr>
<td>0 (%esp)</td>
<td></td>
</tr>
<tr>
<td>Argument count</td>
<td></td>
</tr>
<tr>
<td>Undefined</td>
<td></td>
</tr>
</tbody>
</table>
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.

%esp
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 %esp. 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.
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>a_un</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT_NULL</td>
<td>0</td>
<td>ignored</td>
</tr>
<tr>
<td>AT_IGNORE</td>
<td>1</td>
<td>ignored</td>
</tr>
<tr>
<td>AT_EXECFD</td>
<td>2</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_PHDR</td>
<td>3</td>
<td>a_ptr</td>
</tr>
<tr>
<td>AT_PHENT</td>
<td>4</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_PNUM</td>
<td>5</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_PAGESZ</td>
<td>6</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_BASE</td>
<td>7</td>
<td>a_ptr</td>
</tr>
<tr>
<td>AT_FLAGS</td>
<td>8</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_ENTRY</td>
<td>9</td>
<td>a_ptr</td>
</tr>
<tr>
<td>AT_LIBPATH</td>
<td>10</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_FPHW</td>
<td>11</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_INTP_DEVICE</td>
<td>12</td>
<td>a_val</td>
</tr>
<tr>
<td>AT_INTP_INODE</td>
<td>13</td>
<td>a_val</td>
</tr>
</tbody>
</table>

AT_NULL     The auxiliary vector has no fixed length; instead its last entry's a_type member has this value.
AT_IGNORE  This type indicates the entry has no meaning. The corresponding value of a_un is undefined.

AT_EXECFD  As Chapter 5 describes, exec(BA_OS) may pass control to an 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.

AT_PHNUM  The a_val member of this entry holds the number of entries in the program header table to which the AT_PHDR entry points.

AT_PAGESZ  If present, this entry’s a_val member gives the system page size, 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.
AT_FPHW

The a_val member of this entry will be set to

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>if no floating point support exists</td>
</tr>
<tr>
<td>1</td>
<td>if floating point software emulation exists</td>
</tr>
<tr>
<td>2</td>
<td>if it has a 80287 chip</td>
</tr>
<tr>
<td>3</td>
<td>if it has a 80387 chip or a 80487 chip</td>
</tr>
</tbody>
</table>

Figure 3-34: AT_FPHW values

AT_INTPDEVICE

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.
## Figure 3-35: Example Process Stack

<table>
<thead>
<tr>
<th>n</th>
<th>\0</th>
<th>pad</th>
<th>High addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>\b</td>
<td>i</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>\u</td>
<td>s</td>
<td></td>
</tr>
</tbody>
</table>

0x8047ff0
PATH

dir \0

gome /

E = / h

0x8047fe0
\0 HOME
\0 ABI

ec ho

0x8047fd0
0
0
13
2
0

0x8047fc0
0x8047ff0
0x8047fe1
0
0x8047fdd

0x8047fb0
0x8047fd8

0(%esp), 0x8047fac

2

Undefined

Low addresses

<table>
<thead>
<tr>
<th>Auxiliary vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8047fd0</td>
</tr>
<tr>
<td>0x8047fc0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8047ff0</td>
</tr>
<tr>
<td>0x8047fe1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8047fb0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8047fd8</td>
</tr>
<tr>
<td>0x8047fdd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Undefined</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8047fac</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0(%esp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x8047fac</td>
</tr>
</tbody>
</table>
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.

**NOTE**
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.

**NOTE**
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.
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.

\texttt{name@GOT (@ebx)}

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

\texttt{name@GOTOFF (@ebx)}

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

Coding Examples
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 %ebx to the global offset table’s address. Because %ebx 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

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>call .L1</td>
</tr>
<tr>
<td>2</td>
<td>.L1: popl %ebx</td>
</tr>
<tr>
<td>3</td>
<td>addl <strong>GLOBAL_OFFSET_TABLE</strong>+[.-.L1], %ebx</td>
</tr>
</tbody>
</table>

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 %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.
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 .L1
.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.
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.

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.
Figure 3-40: Position-Independent Static Data Access

C

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

*ptr = src;

Assembly

leal ptr@GOTOFF(%ebx), %eax
leal dst@GOTOFF(%ebx), %edx
movl %edx, (%eax)

movl ptr@GOTOFF(%ebx), %eax
movl src@GOTOFF(%ebx), %edx
movl %edx, (%eax)

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

extern void function();
function();

Assembly

.globl 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.
Figure 3-42: Position-Independent Direct Function Call

C

extern void function();
function();

Assembly
.globl function
.call function@PLT

Indirect function calls use the indirect call instruction.

Figure 3-43: Absolute Indirect Function Call

C

extern void (*ptr)();
extern void name();
ptr = name;
(*ptr)();

Assembly
.globl ptr, name;
movl $name, ptr
call *ptr

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

extern void (*ptr)();
extern void name();
ptr = name;
(*ptr)();

Assembly
.globl ptr, name
movl ptr@GOT(%ebx), %eax
movl name@GOT(%ebx), %edx
movl %edx, (%eax)
movl ptr@GOT(%ebx), %eax
call *(%eax)
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

<table>
<thead>
<tr>
<th>C</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>label:</td>
<td>.L01:</td>
</tr>
<tr>
<td></td>
<td>. . .</td>
</tr>
<tr>
<td>goto label;</td>
<td>jmp .L01</td>
</tr>
</tbody>
</table>

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

**C**

```c
switch (j)
{
    case 0:
    ....
    case 2:
    ....
    case 3:
    ....
    default:
    ....
}
```

**Assembly**

```assembly
cmp $3, %eax
ja .Ldef
jmp * .Ltab(%eax, 4)
.Ltab: .long .Lcase0
.long .Ldef
.long .Lcase2
.long .Lcase3
```

### Figure 3-47: Position-Independent switch Code

**C**

```c
switch (j)
{
    case 0:
    ....
    case 2:
    ....
    case 3:
    ....
    default:
    ....
}
```

**Assembly**

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

<table>
<thead>
<tr>
<th>Base</th>
<th>Offset</th>
<th>Contents</th>
<th>High addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>%ebp</td>
<td>4n+8</td>
<td>argument word n</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>argument word 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>return address</td>
<td></td>
</tr>
<tr>
<td>%ebp</td>
<td>0</td>
<td>caller’s %ebp</td>
<td></td>
</tr>
<tr>
<td>%ebp</td>
<td>-4</td>
<td>x words local space:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>automatic variables,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>temporaries, etc.</td>
<td></td>
</tr>
<tr>
<td>%ebp</td>
<td>-4x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%esp</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%esp</td>
<td>8</td>
<td>caller’s %edi</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>caller’s %esi</td>
<td></td>
</tr>
<tr>
<td>%esp</td>
<td>0</td>
<td>caller’s %ebx</td>
<td></td>
</tr>
</tbody>
</table>

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 %ebp. Its incoming arguments reside in the previous frame, referenced as positive offsets from %ebp. If necessary, a function saves the values of %edi, %esi, and %ebx 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.
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 `<varargs.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 `%ebp` from the area referenced from `%esp`. 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.
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(3)`]. 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**

<table>
<thead>
<tr>
<th>Original</th>
<th>Intermediate</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (%ebp)</td>
<td>arguments and automatic variables</td>
<td>0 (%ebp)</td>
</tr>
<tr>
<td>12 (%esp)</td>
<td>save area 3 words</td>
<td>old save area 3 words</td>
</tr>
<tr>
<td>0 (%esp)</td>
<td><code>undefined</code></td>
<td>12 (%esp)</td>
</tr>
<tr>
<td></td>
<td><code>new space 52 bytes</code></td>
<td><code>new space 52 bytes</code></td>
</tr>
<tr>
<td></td>
<td><code>++++++++++</code></td>
<td><code>++++++++++</code></td>
</tr>
<tr>
<td></td>
<td><code>undefined</code></td>
<td>save area 3 words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 (%esp)</td>
</tr>
</tbody>
</table>

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.
4 OBJECT FILES

- ELF Header
  - Machine Information

- Sections
  - Special Sections

- Symbol Table
  - Symbol Values

- Relocation
  - Relocation Types
ELF Header

Machine Information

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

**Figure 4-1: Intel386 Identification, _e_ident_**

<table>
<thead>
<tr>
<th>Position</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>e_ident[EI_CLASS]</em></td>
<td>ELFCLASS32</td>
</tr>
<tr>
<td><em>e_ident[EI_DATA]</em></td>
<td>ELFDATA2LSB</td>
</tr>
</tbody>
</table>

Processor identification resides in the ELF header’s _e_machine_ member and must have the value _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.
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.got</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC + SHF_WRITE</td>
</tr>
<tr>
<td>.plt</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC + SHF_EXECINSTR</td>
</tr>
</tbody>
</table>

This section holds the global offset table. See “Coding Examples” in Chapter 3 and “Global Offset Table” in Chapter 5 for more information.

This section holds the procedure linkage table. See “Procedure Linkage Table” in Chapter 5 for more information.
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.
Relocation

Relocation Types

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

**Figure 4-3: Relocatable Fields**

![Fig. 4-3: Relocatable Fields](image)

<table>
<thead>
<tr>
<th>word32</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01020304</td>
<td><img src="image" alt="Hexadecimal value" /></td>
</tr>
</tbody>
</table>

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.
GOT  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.

L  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.

P  This means the place (section offset or address) of the storage unit being relocated (computed using r_offset).

S  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 Elf32_Rel relocation entries, the field to be relocated holds the addend. In all cases, the addend and the computed result use the same byte order.

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.

---

**Figure 4-4: Relocation Types**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Field</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_386_NONE</td>
<td>0</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>R_386_32</td>
<td>1</td>
<td>word32</td>
<td>( S + A )</td>
</tr>
<tr>
<td>R_386_PC32</td>
<td>2</td>
<td>word32</td>
<td>( S + A - P )</td>
</tr>
<tr>
<td>R_386_GOT32</td>
<td>3</td>
<td>word32</td>
<td>( G + A - P )</td>
</tr>
<tr>
<td>R_386_PLT32</td>
<td>4</td>
<td>word32</td>
<td>( L + A - P )</td>
</tr>
<tr>
<td>R_386_COPY</td>
<td>5</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>R_386_GLOB_DAT</td>
<td>6</td>
<td>word32</td>
<td>( S )</td>
</tr>
<tr>
<td>R_386_JMP_SLOT</td>
<td>7</td>
<td>word32</td>
<td>( S )</td>
</tr>
<tr>
<td>R_386_RELATIVE</td>
<td>8</td>
<td>word32</td>
<td>( B + A )</td>
</tr>
<tr>
<td>R_386_GOTOFF</td>
<td>9</td>
<td>word32</td>
<td>( S + A - \text{GOT} )</td>
</tr>
<tr>
<td>R_386_GOTPC</td>
<td>10</td>
<td>word32</td>
<td>( \text{GOT} + A - P )</td>
</tr>
</tbody>
</table>

---
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.

R_386_COPY  The link editor creates this relocation type for dynamic linking. 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.

R_386_GLOB_DAT  This relocation type is used to set a global offset table entry to 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.

R_386_GOTPC  This relocation type resembles R_386_PC32, except it uses the 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.
# 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
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 ($0 \times 1000$) 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.

**Figure 5-1: Executable File**

<table>
<thead>
<tr>
<th>File Offset</th>
<th>File</th>
<th>Virtual Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ELF header</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Program header table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other information</td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>Text segment</td>
<td>0x8048100</td>
</tr>
<tr>
<td></td>
<td>0x2be00 bytes</td>
<td>0x8073eff</td>
</tr>
<tr>
<td>0x2bf00</td>
<td>Data segment</td>
<td>0x8074f00</td>
</tr>
<tr>
<td></td>
<td>0x4e00 bytes</td>
<td>0x8079cff</td>
</tr>
<tr>
<td>0x30d00</td>
<td>Other information</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
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 KB (0x1000) pages.

### Figure 5-2: Program Header Segments

<table>
<thead>
<tr>
<th>Member</th>
<th>Text</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_type</td>
<td>PT_LOAD</td>
<td>PT_LOAD</td>
</tr>
<tr>
<td>p_offset</td>
<td>0x100</td>
<td>0x2bf00</td>
</tr>
<tr>
<td>p_vaddr</td>
<td>0x8048100</td>
<td>0x8074f00</td>
</tr>
<tr>
<td>p_paddr</td>
<td>unspecified</td>
<td>unspecified</td>
</tr>
<tr>
<td>p_filesz</td>
<td>0x2be00</td>
<td>0x4e00</td>
</tr>
<tr>
<td>p_memsz</td>
<td>0x2be00</td>
<td>0x5e24</td>
</tr>
<tr>
<td>p_flags</td>
<td>PF_R+PF_X</td>
<td>PF_R+PF_W+PF_X</td>
</tr>
<tr>
<td>p_align</td>
<td>0x1000</td>
<td>0x1000</td>
</tr>
</tbody>
</table>

DRAFT COPY
March 19, 1997
File: abi_386/chap5 (Delta 44.3)
386:adm.book:sum
Page: 82
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.
Figure 5-4: Example Shared Object Segment Addresses

<table>
<thead>
<tr>
<th>Source</th>
<th>Text</th>
<th>Data</th>
<th>Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>0x200</td>
<td>0x2a400</td>
<td>0x0</td>
</tr>
<tr>
<td>Process 1</td>
<td>0x80000200</td>
<td>0x8002a400</td>
<td>0x80000000</td>
</tr>
<tr>
<td>Process 2</td>
<td>0x80081200</td>
<td>0x800ab400</td>
<td>0x80081000</td>
</tr>
<tr>
<td>Process 3</td>
<td>0x900c0200</td>
<td>0x900ea400</td>
<td>0x900c0000</td>
</tr>
<tr>
<td>Process 4</td>
<td>0x900c6200</td>
<td>0x900f0400</td>
<td>0x900c6000</td>
</tr>
</tbody>
</table>
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.
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.

```
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
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.

1. 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.
Figure 5-6: Absolute Procedure Linkage Table

```
.PLT0: pushl  got_plus_4
    jmp *got_plus_8
    nop; nop
    nop; nop

.PLT1: jmp *name1_in_GOT
    pushl $offset
    jmp .PLT0@PC

.PLT2: jmp *name2_in_GOT
    pushl $offset
    jmp .PLT0@PC

...
```

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

...
```
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 pushl 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 .PLT0, 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.
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 `pushl` 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**

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

```
/usr/lib/libc.so.1
```
# 6 LIBRARIES

## Shared Library Names

- C Library
  - Additional Entry Points
  - Support Routines
  - Global Data Symbols
    - Application Constraints

## System Data Interfaces

- Data Definitions
  - Reentrancy Considerations
- X Window Data Definitions
- Motif 1.2 Data Definitions
- TCP/IP Data Definitions
Shared Library Names

The version number of the libraries named in the System V Generic ABI is specified below.

Figure 6-1: Shared Library Names

<table>
<thead>
<tr>
<th>Library Reference Name</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>libc.so.1</td>
<td>M</td>
</tr>
<tr>
<td>libthread.so.1</td>
<td>M</td>
</tr>
<tr>
<td>libdl.so.1</td>
<td>M</td>
</tr>
<tr>
<td>libnsl.so.1</td>
<td>M</td>
</tr>
<tr>
<td>libX11.so.5.0</td>
<td>M</td>
</tr>
<tr>
<td>libXt.so.5.0</td>
<td>M</td>
</tr>
<tr>
<td>libXext.so.5.0</td>
<td>M</td>
</tr>
<tr>
<td>libXm.so.1.2</td>
<td>M</td>
</tr>
<tr>
<td>libMrM.so.1.2</td>
<td>M</td>
</tr>
</tbody>
</table>
C Library

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 summary for each function follows the table.

Figure 6-2: libc Additional Required Entry Points

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_fxstat</td>
<td>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.</td>
</tr>
<tr>
<td>_lxstat</td>
<td>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.</td>
</tr>
<tr>
<td>_xstat</td>
<td>The semantics of this function are identical to those of the stat(BA_OS) function described in the System V Interface Definition, Edition 4.</td>
</tr>
<tr>
<td>_xmknod</td>
<td>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.</td>
</tr>
<tr>
<td>_xstat</td>
<td>The semantics of this function are identical to those of the stat(BA_OS) function described in the System V Interface Definition, Edition 4.</td>
</tr>
</tbody>
</table>

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

Support Routines

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

Figure 6-3: libc, Support Routines

<table>
<thead>
<tr>
<th>_fpstart</th>
<th>_fpstart</th>
<th>sbrk</th>
<th>_sbrk</th>
</tr>
</thead>
</table>

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, sbrk returns the old break value. Otherwise, it returns -1 and sets errno to indicate the error. The symbol _sbrk 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.
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.
```
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 specified 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.

Reentrancy Considerations

New conventions have been added to accommodate 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.

NOTE

All information presented in the figures marked with * are new to the Fourth Edition of the psABI.
Figure 6-5: <aio.h>*

```c
struct aiocb {
    int aifoildes;
    volatile void* aio_buf;
    size_t aio_nbytes;
    off_t aio_offset;
    int aio_recprio;
    struct sigevent aio_sigevent;
    int 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>

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

CAUTION

System Data Interfaces 6-7
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;
   char   d_name[1];
};

Figure 6-9: <dlfcn.h>*

#define RTLD_LAZY   1
#define RTLD_NOW    2
#define RTLD_GLOBAL 4
```c
#define ELF32_FSZ_ADDR 4
#define ELF32_FSZ_HALF 2
#define ELF32_FSZ_OFF 4
#define ELF32_FSZ_SWORD 4
#define ELF32_FSZ_WORD 4

#define EI_NIDENT 16

typedef struct {
    unsigned char _ident[EI_NIDENT];
    Elf32_Half e_type;
    Elf32_Half e_machine;
    Elf32_Word e_version;
    Elf32_Addr e_entry;
    Elf32_Off e_poff;
    Elf32_Off e_shoff;
    Elf32_Word e_flags;
    Elf32_Half e_ehsize;
    Elf32_Half e_version;
    Elf32_Half e_phentsize;
    Elf32_Half e_phnum;
    Elf32_Half e_shentsize;
    Elf32_Half e_shnum;
    Elf32_Half e_shstrndx;
} Elf32_Ehdr;

#define ELF_MAG0 0x7f
#define ELF_MAG1 'E'
#define ELF_MAG2 'L'
#define ELF_MAG3 'F'
#define ELF_MAG "\\177ELF"
#define SELFMAG 4
```


```c
#define EI_MAG0 0
#define EI_MAG1 1
#define EI_MAG2 2
#define EI_MAG3 3
#define EI_CLASS 4
#define EI_DATA 5
#define EI_VERSION 6
#define EI_PAD 7

#define ELFCLASSNONE 0
#define ELFCLASS32 1
#define ELFCLASS64 2
#define ELFCLASSNUM 3
#define ELFDATAONE 0
#define ELFDATANONE 1
#define ELFDATA2LSB 2
#define ELFDATA2MSB 3

#define ET_NONE 0
#define ET_REL 1
#define ET_EXEC 2
#define ET_DYN 3
#define ET_CORE 4
#define ET_NUM 5

#define ET_LOPROC 0xff00
#define ET_HIPROC 0xffff

#define EM_NONE 0
#define EM_M32 1
#define EM_SPARC 2
#define EM_386 3
#define EM_68K 4
#define EM_88K 5
#define EM_486 6
#define EM_860 7
#define EM_NUM 8
```
#define EV_NONE 0
#define EV_CURRENT 1
#define EV_NUM 2

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
#define PT_SHLIB 5
#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
Figure 6-13: `<elf.h>`, Part 4 of 6

```c
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;
    Elf32_Word sh_link;
    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 2
#define SHT_STRTAB 3
#define SHT_RELA 4
#define SHT_HASH 5
#define SHT_DYNAMIC 6
#define SHT_NOTE 7
#define SHT_NOBITS 8
#define SHT_REL 9
#define SHT_SHLIB 10
#define SHT_DYNSYM 11
#define SHT_NUM 12

#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_EXECDYN 0x4
```

6-12 LIBRARIES
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 STT_NOTYPE 0
#define STT_OBJECT 1
#define STT_FUNC 2
#define STT_SECTION 3
#define STT_FILE 4
#define STT_NUM 5
#define STT_LPROC 13
#define STT_HPROC 15

typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
} Elf32_Rel;

typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
    Elf32_Word 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))
# define EPERM 1
# define ENOENT 2
# define ESRCH 3
# define EINTR 4
# define EO 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
# defineEFAULT 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
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 ENODATA   63
#define ENONET    64
#define ENOPKG    65
#define EREMOTE   66
#define ENOLINK   67
```
Figure 6-18: `<errno.h>`, Part 3 of 3

```c
#define EADV 68
#define ESRMNT 69
#define ECOMM 70
#define EPROTO 71
#define EMULTIHWOP 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 ESTPIPE 92
#define ENOTEMPTY 93
#define E USERS 94
#define ECONNABORTED 130 G
#define CONNRESET 131 G
#define ECONNREFUSED 146 G
#define EINPROGRESS 150 M
#define ESTALE 151
#define ECANCELED 158 M

/* Non-reentrant */
extern int errno;

/* Reentrant */
#define errno (*__thr_errno())
```
Figure 6-19: `<fcntl.h>`, Part 1 of 2

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

```c
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

```c
extern int __flt_rounds;
#define FLT_ROUNDS __flt_rounds
#define FLT_RADIX 2
#define FLT_MANT_DIG 24
#define FLT_EPSILON 1.19209290E-07F
#define FLT_DIG 6
#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

```c
#define DBL_MANT_DIG 53
#define DBL_EPSILON 2.2204460492503131E-16
#define DBL_DIG 15
#define DBL_MIN_EXP (-1021)
#define DBL_MIN 2.2250738585072014E-308
#define DBL_MIN_10_EXP (-307)
#define DBL_MAX_EXP (+1024)
#define DBL_MAX 1.7976931348623157E+308
#define DBL_MAX_10_EXP (+308)
```
Figure 6-23: <float.h>, Extended-Precision

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

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

```c
#define MM_NULL 0L
#define MM_HARD 0x00000001L
#define MM_SOFT 0x00000002L
#define MM_FIRM 0x00000004L
#define MM_RECOVER 0x00000100L
#define MM_NRECov 0x00000200L
#define MM_APPL 0x00000008L
#define MM_UTIL 0x00000010L
#define MM_OPSYS 0x00000020L
#define MM_PRINT 0x00000040L
#define MM_CONSOLE 0x00000080L
```
Figure 6-25: <fmtmsg.h>, Part 2 of 2

```c
#define MM_NOSEV 0
#define MM_HALT 1
#define MM_ERROR 2
#define MM_WARNING 3
#define MM_INFO 4

#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 -1
#define MM_OK 0x00
#define MM_NOMSG 0x01
#define MM_NOCON 0x04
```

Figure 6-26: <fmatch.h>*

```c
#define FNM_PATHNAME 0x001
#define FNM_PERIOD 0x002
#define FNM_NOESCAPE 0x004
#define FNM_BADRANGE 0x008
#define FNM_EXTENDED 0x020

#define FNM_NOSYS (-1)
#define FNM_NOMATCH (-2)
```
Figure 6-27: <ftw.h>

```c
#define FTW_F 0
#define FTW_D 1
#define FTW_DNR 2
#define FTW_NS 3
#define FTW_SL 4
#define FTW_DP 6
#define FTW_SLN 7
#define FTW_PHYS 01
#define FTW_MOUNT 02
#define FTW_CHDIR 04
#define FTW_DEPTH 010

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>*

```c
#define GLOB_APPEND 0x0001
#define GLOB_DOFFS 0x0002
#define GLOB_ERR 0x0004
#define GLOB_MARK 0x0008
#define GLOB_NPCHECK 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;
    size_t     gl_offs;
} glob_t;
```
Figure 6-29: <grp.h>

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

Figure 6-30: <iconv.h>*

```c
typedef void *iconv_t;
```


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

```c
struct ipc_perm {
    uid_t uid;
    gid_t gid;
    uid_t cuid;
    gid_t cgid;
    mode_t mode;
    ulong seq;
    key_t key;
    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
```
### Figure 6-32: `<langinfo.h>`, Part 1 of 2

<table>
<thead>
<tr>
<th>Define</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAY_1</td>
<td>1</td>
</tr>
<tr>
<td>DAY_2</td>
<td>2</td>
</tr>
<tr>
<td>DAY_3</td>
<td>3</td>
</tr>
<tr>
<td>DAY_4</td>
<td>4</td>
</tr>
<tr>
<td>DAY_5</td>
<td>5</td>
</tr>
<tr>
<td>DAY_6</td>
<td>6</td>
</tr>
<tr>
<td>DAY_7</td>
<td>7</td>
</tr>
<tr>
<td>ABDAY_1</td>
<td>8</td>
</tr>
<tr>
<td>ABDAY_2</td>
<td>9</td>
</tr>
<tr>
<td>ABDAY_3</td>
<td>10</td>
</tr>
<tr>
<td>ABDAY_4</td>
<td>11</td>
</tr>
<tr>
<td>ABDAY_5</td>
<td>12</td>
</tr>
<tr>
<td>ABDAY_6</td>
<td>13</td>
</tr>
<tr>
<td>ABDAY_7</td>
<td>14</td>
</tr>
<tr>
<td>MON_1</td>
<td>15</td>
</tr>
<tr>
<td>MON_2</td>
<td>16</td>
</tr>
<tr>
<td>MON_3</td>
<td>17</td>
</tr>
<tr>
<td>MON_4</td>
<td>18</td>
</tr>
<tr>
<td>MON_5</td>
<td>19</td>
</tr>
<tr>
<td>MON_6</td>
<td>20</td>
</tr>
<tr>
<td>MON_7</td>
<td>21</td>
</tr>
<tr>
<td>MON_8</td>
<td>22</td>
</tr>
<tr>
<td>MON_9</td>
<td>23</td>
</tr>
<tr>
<td>MON_10</td>
<td>24</td>
</tr>
<tr>
<td>MON_11</td>
<td>25</td>
</tr>
<tr>
<td>MON_12</td>
<td>26</td>
</tr>
</tbody>
</table>
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 M
#define T_FMT_AMPM 50 M
#define ERA 51 M
#define ERA_D_FMT 52 M
#define ERA_D_T_FMT 53 M
#define ERA_T_FMT 54 M
#define ALT_DIGITS 55 M
#define YESEXPR 56 M
#define NOEXPR 57 M
Figure 6-34: <limits.h>, Part 1 of 2

```c
#define CHAR_BIT 8
#define SCHAR_MIN (-128)
#define SCHAR_MAX 127
#define UCHAR_MAX 255
#define MB_LEN_MAX 5

#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: `<limits.h>`, Part 2 of 2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NL_ARGMAX</td>
<td>9</td>
</tr>
<tr>
<td>NL_LANGMAX</td>
<td>14</td>
</tr>
<tr>
<td>NL_MSGMAX</td>
<td>32767</td>
</tr>
<tr>
<td>NL_NMAX</td>
<td>1</td>
</tr>
<tr>
<td>NL_SETMAX</td>
<td>255</td>
</tr>
<tr>
<td>NL_TEXTMAX</td>
<td>255</td>
</tr>
<tr>
<td>NZERO</td>
<td>20</td>
</tr>
<tr>
<td>WORD_BIT</td>
<td>32</td>
</tr>
<tr>
<td>LONG_BIT</td>
<td>32</td>
</tr>
<tr>
<td>DBL_DIG</td>
<td>15</td>
</tr>
<tr>
<td>DBL_MAX</td>
<td>1.7976931348623157E+308</td>
</tr>
<tr>
<td>DBL_MIN</td>
<td>2.2250738585072014E-308</td>
</tr>
<tr>
<td>FLT_DIG</td>
<td>6</td>
</tr>
<tr>
<td>FLT_MAX</td>
<td>3.40282347E+38F</td>
</tr>
<tr>
<td>FLT_MIN</td>
<td>1.17549435E-38F</td>
</tr>
<tr>
<td>FCHR_MAX</td>
<td>1048576</td>
</tr>
</tbody>
</table>
Figure 6-36: `<locale.h>`

```
struct lconv {
    char *decimal_point;
    char *thousands_sep;
    char *grouping;
    char *int_curr_symbol;
    char *currency_symbol;
    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;
};
```

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

typedef volatile struct {
    char    wanted;
} lwp_cond_t;

typedef volatile unsigned char _simplelock_t;

extern const double __huge_val;
#define HUGE_VAL     __huge_val
Figure 6-40: <sys/mman.h>

```c
#define PROT_READ 0x1
#define PROT_WRITE 0x2
#define PROT_EXEC 0x4
#define PROT_NONE 0x0

#define MAP_SHARED 1
#define MAP_PRIVATE 2
#define MAP_FIXED 0x10

#define MS_SYNC 0x0
#define MS_ASYNC 0x1
#define MS_INVALIDATE 0x2

#define PROC_TEXT (PROT_EXEC | PROT_READ)
#define PROC_DATA (PROT_READ | PROT_WRITE | PROT_EXEC)

#define SHARED 0x10
#define PRIVATE 0x20

#define MC_SYNC 1
#define MC_LOCK 2
#define MC_UNLOCK 3
#define MC_LOCKAS 5
#define MC_UNLOCKAS 6

#define MCL_CURRENT 0x1
#define MCL_FUTURE 0x2
```
#define VOID void
#define MAXPATHLEN 1024
#define MODMAXLINKINFOLEN 32

struct modspecific_stat {
    char mss_linkinfo[MODMAXLINKINFOLEN];
    int mss_type;
    int mss_p0[2];
    int mss_pl[2];
};

#define MODMAXLINK 4

struct modstatus {
    int ms_id;
    VOID *ms_base;
    unsigned int ms_size;
    int ms_rev;
    char ms_path[MAXPATHLEN];
    time_t ms_unload_delay;
    int ms_refcnt;
    int ms_depcnt;
    struct modspecific_stat ms_msinfo[MODMAXLINK];
};
Figure 6-42: <sys/mount.h>

```c
#define MS_RDONLY 0x01
#define MS_FSS 0x02
#define MS_DATA 0x04
#define MS_HADBAD 0x08
#define MS_NOSUID 0x10
#define MS_REMOUNT 0x20
#define MS_NOTRUNC 0x40
```
Figure 6-43: <sys/msg.h>

```c
#define MSG_NOERROR 010000

struct msqid_ds {
    struct ipc_perm msg_perm;
    struct msg *msg_first;
    struct msg *msg_last;
    ulong msg_cbytes;
    ulong msg_qnum;
    ulong msg_qbytes;
    pid_t msg_lspid;
    pid_t msg_lrpid;
    time_t msg_stime;
    long msg_pad1;
    time_t msg_rtime;
    long msg_pad2;
    time_t msg_ctime;
    long msg_pad3;
    long msg_pad4[4];
};

struct msg {
    struct msg *msg_next;
    long msg_type;
    ushort msg_ts;
    short msg_spot;
};
```
Figure 6-44: \texttt{<netconfig.h>}, Part 1 of 2

```c
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_nlookup;
    char **nc_lookups;
    unsigned long nc_unused[8];
};

#define NC_TPI_CLTS 1
#define NC_TPI_COTS 2
#define NC_TPI_COTS_ORD 3
#define NC_TPI_RAW 4

#define NC_NOFLAG 00
#define NC_VISIBLE 01
```
Figure 6-45: `<netconfig.h>`, Part 2 of 2

```c
#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_APPLEALK "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

```c
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      0
#define ND_HOSTSERVLIST  1
#define ND_ADDR          2
#define ND_ADDRLIST      3

#define ND_BADARG        -2
#define ND_NOMEM         -1
#define ND_OK            0
#define ND_NOHOST        1
#define ND_NOSERV        2
#define ND_NOSYM         3
#define ND_OPEN          4
#define ND_ACCESS        5
#define ND_UNKNOWN       6
#define ND_NOCTRL        7
#define ND_FAILCTRL      8
#define ND_SYSTEM        9
#define ND_NOERRMEM      10
#define ND_NOLIB         11
#define ND_XTIERRO       12
#define ND_BADSTATE      13
```
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;
```
Figure 6-49: <sys/param.h>

#define CANBSIZ  256
#define HZ      100
#define TICK    10000000
#define NGROUPS_UMIN 0
#define NBPSCTR  512
#define MAXPATHLEN 1024
#define MAXSYMLINKS 20
#define MAXNAMELEN 256
#define NADDR    13
#define PIPE_MAX 5120
#define NBBY     8
#define MAXFRAG  8
Figure 6-50: `<poll.h>`

```c
struct pollfd {
    int fd;
    short events;
    short revents;
};

#define POLLIN 0x0001
#define POLLPRI 0x0002
#define POLLOUT 0x0004
#define POLLRDERROR 0x40
#define POLLWRNORM POLLOUT
#define POLLRDDBAND 0x0080
#define POLLWRBAND 0x0100

#define POLLNORM PollRDNorm
#define POLLERR 0x0008
#define POLLPOLLHUP 0x0010
#define POLLNVAL 0x0020
```
**Figure 6-51:** `<sys/priocntl.h>`

```c
#define PC_GETCID 0
#define PC_GETCLINFO 1
#define PC_SETPARMS 2
#define PC_GETPARMS 3

#define PC_CLNNULL -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: \texttt{<sys/procset.h>}

```c
#define P_INITPID 1
#define P_INITUID 0
#define P_INITPGID 0

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;
    idtype_t p_ridtype;
    id_t p_rid;
} procset_t;
```
Figure 6-53: <pwd.h>

```c
struct passwd {
    char    *pw_name;
    char    *pw_passwd;
    uid_t   pw_uid;
    gid_t   pw_gid;
    char    *pw_age;
    char    *pw_comment;
    char    *pw_gecos;
    char    *pw_dir;
    char    *pw_shell;
};
```
```c
#define REG_NOTBOL 0x000001
#define REG_NOTEOL 0x000002
#define REG_NONEMPTY 0x000004

#define REG_OR 0x000001
#define REG_PLUS 0x000002
#define REGQUEST 0x000004
#define REG_BRACES 0x000008
#define REG_PARENS 0x000010
#define REG_ANCHORS 0x000020
#define REG_NOBACKREF 0x000040
#define REG_NOAUTOQUOTE 0x000080

#define REG_EXTENDED (REG_OR | REG_PLUS | REGQUEST |
                    REG_BRACES | REG_PARENS | REG_ANCHORS |
                    REG_NOBACKREF | REG_NOAUTOQUOTE)
#define REG_ICASE 0x000100
#define REG_NOSUB 0x000200
#define REG_NEWLINE 0x000400
#define REG_ONESUB 0x000800
#define REG_BADRANGE 0x004000
#define REGANGLES 0x040000
#define REG_ESCNL 0x080000
#define REG_OLDDBRE (REG_BADRANGE | REGANGLES | REG_ESCNL)
```
Figure 6-55: <regex.h>* , Part 2 of 2

```c
#define REG_ENOSYS (-1)
#define REG_NOMATCH 1
#define REG_BADPAT 2
#define REG_ECOLLATE 3
#define REG_ECTYPE 4
#define REG_EESCAPE 7
#define REG_PSUBREG 8
#define REG_EBRACK 9
#define REG_NOFAT 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;
    void *[4];
} 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>`

```c
#define RLIMIT_CPU 0
#define RLIMITFSIZE 1
#define RLIMITDATA 2
#define RLIMITSTACK 3
#define RLIMITCORE 4
#define RLIMITNOFILE 5
#define RLIMIT_VMEM 6
#define RLIM_NLIMITS 7
#define RLIMIT_AS RLIMIT_VMEM
#define RLIMIT_INFINITY 0x7fffffff

typedef unsigned long rlim_t;

struct rlimit {
    rlim_t rlim_cur;
    rlim_t rlim_max;
};
```
```c
#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_putchar)();
        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;
```
# define xdr_getpos(xdrs) \
  (*(xdrs)->x_ops->x_getposn)(xdrs)
#define xdr_setpos(xdrs, pos) \
  (*(xdrs)->x_ops->x_setposn)(xdrs, pos)
#define xdr_inline(xdrs, len) \
  (*(xdrs)->x_ops->x_inline)(xdrs, len)
#define xdr_destroy(xdrs) \
  (*(xdrs)->x_ops->x_destroy)(xdrs)

#define NULL_xdrproc_t ((xdrproc_t)0)
struct xdr_discrim {
  int     value;
  xdrproc_t  proc;
};
Figure 6-59: <rpc.h>, Part 3 of 16

```c
#define MAX_AUTH_BYTES 400
#define MAX_NETNAMELEN 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 u_int32
union des_block {
    struct {
        u_int32   high;
        u_int32   low;
    } key;
    char      c[8];
};
typedef union des_block des_block;
```
```c
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_marshall)();
        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    3
#define AUTH_ESV    200004
```
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_PROGVERS mismatch=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_NZAXLATEFAILURE=22,
    RPC_TLIERROR=20,
    RPC_FAILED=16
};
#define RPC_PMAPFAILURE RPC_RPCBFailure
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;
            u_long high;
        } 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;
Figure 6-63: <rpc.h>, Part 7 of 16

```c
#define FEEDBACK_REXMIT1 1
#define FEEDBACK_OK 2

#define cint_call(rh, proc, xargs, argsp, xres, resp, secs) \    (*)(rh)->cl_ops->cl_call \    (rh, proc, xargs, argsp, xres, resp, secs)
#define cint_abort(rh) \    (*)(rh)->cl_ops->cl_abort(rh)
#define cint_geterr(rh, errp) \    (*)(rh)->cl_ops->cl_geterr(rh, errp)
#define cint_freeres(rh, xres, resp) \    (*)(rh)->cl_ops->cl_freeres(rh, xres, resp)
#define cint_control(cl, rq, in) \    (*)(cl)->cl_ops->cl_control(cl, rq, in)
#define cint_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 8
#define CLSET_FD_NCLOSE 9
#define CLSET_RETRY_TIMEOUT 4
#define CLGET_RETRY_TIMEOUT 5
```
Figure 6-64: `<rpc.h>`, Part 8 of 16

typedef struct {
    enum clnt_stat cf_stat;
    struct rpc_err cf_error;
} rpc_createerr_t;

extern rpc_createerr_t rpc_createerr;
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
};
```c
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;
};
```
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        cbVers;
    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: `<rpc.h>`, Part 12 of 16

```c
struct authsys_parms {
    u_long    aup_time;
    char      *aup_machname;
    uid_t     aup_uid;
    gid_t     aup_gid;
    u_int     aup_len;
    gid_t     *aup_gids;
};
```

Figure 6-69: `<rpc.h>`, Part 13 of 16

```c
enum authdes_namekind {
    ADN_FULLNAME,
    ADN_NICKNAME
};

struct authdes_fullname {
    char        *name;
    des_block   key;
    u_long      window;
};

struct authdes_cred {
    enum authdes_namekind    adc_namekind;
    struct authdes_fullname  adc_fullname;
    u_long                   adcNickname;
};
```
enum xprt_stat {
    XPRT_DIED,
    XPRT_MOREREQS,
    XPRT_IDLE
};

typedef struct {
    int          xp_fd;
    u_short      xp_port;
    struct xprt_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_laddr;
    struct netbuf xp_raddr;
    char         xp_addr[16];
    struct opaque_auth xp_verf;
    caddr_t      xp_p1;
    caddr_t      xp_p2;
    caddr_t      xp_p3;
    int          xp_type;
} SVCXprt;
# define svc_getrpccaller (x) (& (x)->xp_rtaddr)
# define svc_getargs (xp, xargs, argsp) \
    /* (xp)->xp_ops->xp_getargs ((xp), (xargs), (argsp))
# define svc_freeargs (xp, xargs, argsp) \
    /* (xp)->xp_ops->xp_freeargs ((xp), (xargs), (argsp))
# define svc_destroy (xp) \
    /* (xp)->xp_ops->xp_destroy (xp)

struct svc_req {
    u_long    rq_prog;
    u_long    rqvers;
    u_long    rq_proc;
    struct opaque_auth rq_cred;
    caddr_t   rq_clntcred;
    SVCXPRT   *rq_xprt;
};

#define FD_SETSIZE 1024   M
#define NBBY 8       M
typedef long    fd_mask;   M
#define NFDBITS (sizeof (fd_mask) * NBBY) M
#define howmany(x, y) (((x)+((y)-1))/y) M

typedef struct fd_set {
    fd_mask fds_bits [howmany (FD_SETSIZE, NFDBITS)]; M
} fd_set; M

extern fd_set svc_fdset;
Figure 6-72: <rpc.h>, Part 16 of 16

```c
struct rpcb {
    u_long    r_prog;
    u_long    r_ver;
    char      *r_netid;
    char      *r_addr;
    char      *r_owner;
};
typedef struct rpcb RPCB;

struct rpcblist {
    RPCB          rpcb_map;
    struct rpcblist *rpcb_next;
};
```

Figure 6-73: <rtpriocntl.h>*

```c
typedef struct rtparms {
    short rt_pri;
    ulong rt_qsecs;
    long  rt_tqseccs;
} rtparms_t;

typedef struct rtinfo {
    short rt_maxpri;
} rtinfo_t;

#define RT_NOCHANGE -1
#define RT_TQINF    -2
#define RT_TQDEF   -3
```
typedef enum { FIND, ENTER } ACTION;
typedef struct entry { char *key; void *data; } ENTRY;
typedef enum { preorder, postorder, endorder, leaf } VISIT;
# define SEM_UNDO 010000
# define GETNCNT 3
# define GETPID 4
# define GETVAL 5
# define GETALL 6
# 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;
    long sem_pad2;
    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>`

```c
#define _SIGJBLEN 128
#define _JBLEN 10

typedef int jmp_buf[_JBLEN];
typedef int sigjmp_buf[_SIGJBLEN];
```

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

```c
#define SHMLBA  ((1)<<12)
#define SHM_RDONLY  010000
#define SHM_RDWR  020000

struct shmid_ds {
    struct ipc_perm shm_perm;
    int shm_segsz;
    struct anon_map *shm_amp;
    ushort shm_lkcnt;
    pid_t shm_lpid;
    pid_t shm_cpid;
    ulong shm_nattch;
    ulong shm_cnattch;
    time_t shm_atime;
    long shm_pad1;
    time_t shm_dtime;
    long shm_pad2;
    time_t shm_ctime;
    long shm_pad3;
    long shm_pad4[4];
};
```
Figure 6-78: `<signal.h>`, Part 1 of 3

```c
#define SIGHUP 1
#define SIGINT 2
#define SIGQUIT 3
#define SIGILL 4
#define SIGTRAP 5
#define SIGIO 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 SIGHWR 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

```c
#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 1
#define SIG_UNBLOCK 2
#define SIG_SETMASK 3

typedef struct {
    unsigned int sa_sigbits[4];
} sigset_t;

struct sigaction {
    int sa_flags;
    void (*sa_handler)();
    sigset_t sa_mask;
    int sa_resv[2];
};

#define SA_NOCLDSTOP 0x00020000
#define SA_ONSTACK 0x00000001
#define SA_RESETHAND 0x00000002
#define SA_RESTART 0x00000004
#define SA_SIGINFO 0x00000008
#define SA_NODEFER 0x00000010
#define SA_NOCLEANUP 0x00000000
```
Figure 6-80: `<signal.h>`, Part 3 of 3

```c
#define SS_ONSTACK 0x00000001
#define SS_DISABLE 0x00000002

struct sigaltstack {
    char *ss_sp;
    int ss_size;
    int ss_flags;
};

typedef struct sigaltstack stack_t;
```

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

```c
#define ILL_ILLOPC 1
#define ILL_ILLOPN 2
#define ILL_ILLODF 3
#define ILL_ILLTRP 4
#define ILL_FRVOPC 5
#define ILL_FRVREG 6
#define ILL_COPROC 7
#define ILL_BADSTK 8
#define FPE_INTDIV 1
#define FPE_INTOVF 2
#define FPE_FLTDIV 3
#define FPE_FLOVF 4
#define FPE_FLTUND 5
#define FPE_FLTRES 6
#define FPE_FLTINV 7
#define FPE_FLTSUB 8
```
Figure 6-82: `<sys/siginfo.h>`, Part 2 of 5

```c
#define SEGV_MAPERR 1
#define SEGV_ACCERR 2
#define BUS_ADRALN 1
#define BUS_ADRERR 2
#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)
```
typedef struct siginfo {
    int    si_signo;
    int    si_code;
    int    si_errno;
    union {
        int   _pad[Si_PAD];
        struct {
            pid_t  _pid;
            union {
                struct {
                    uid_t   _uid;
                }  _kill;
                struct {
                    clock_t   _utime;
                    int        _status;
                    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

```c
#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
#define si_band      _data.file_band
```

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

```c
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 1
#define SIGEV_SIGNAL 2
#define SIGEV_CALLBACK 3
```
```c
#define _ST_FSTYP_SZ 16

struct stat {
    dev_t st_dev;
    long st_pad1[3];
    ino_t st_ino;
    mode_t st_mode;
    nlink_t st_nlink;
    uid_t st_uid;
    gid_t st_gid;
    dev_t st_rdev;
    long st_pad2[2];
    off_t st_size;
    long st_pad3;
    timespec_t st_atim;
    timespec_t st_mtim;
    timespec_t st_ctim;
    long st_blksize;
    long st_blocks;
    char st_fstype[ST_FSTYP_SZ];
    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
```
```c
#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_IFLINK 0xA000

#define S_ISUID 0x800
#define S_ISGID 0x400
#define S_ISVTX 0x200
#define S_IRWXU 00700
#define S_IRUSR 00400
#define S_IWUSR 00200
#define S_IXUSR 00100
#define S_IRWXG 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)
#define S_ISCHR(mode)     ((mode&0xF000) == 0x2000)
#define S_ISDIR(mode)     ((mode&0xF000) == 0x4000)
#define S_ISBLK(mode)     ((mode&0xF000) == 0x6000)
#define S_ISREG(mode)     ((mode&0xF000) == 0x8000)
```
Figure 6-88: <sys/statvfs.h>

```c
#define FSTYPSZ 16
typedef struct statvfs {
    u_long f_bsize;
    u_long f_frsize;
    u_long f_blocks;
    u_long f_bfree;
    u_long f_bavail;
    u_long f_files;
    u_long f_ffree;
    u_long f_favail;
    u_long f_fsid;
    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>`

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

Figure 6-90: `<stddef.h>`

```c
typedef int ptrdiff_t;
typedef unsigned int size_t;
#define NULL 0
typedef long wchar_t;
#define offsetof(s, m) (size_t)(&((s *)0)->m))
```
typedef unsigned int size_t;
typedef long 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 _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

```c
typedef struct {
    int _cnt;
    unsigned char * _ptr;
    unsigned char * _base;
    unsigned char _flag;
    unsigned char _file; ²²
} FILE;

extern FILE __iob[];

#define clearerr(p) ((void)((p)->_flag & ~(IOERR | IOEOF))) †
#define feof(p) ((p)->_flag & IOEOF)
#define ferror(p) ((p)->_flag & IOERR)
#define fileno(p) (p)->_file ²²

/* Non reentrant */
#define getc_unlocked(p) (-(p)->_cnt < 0 ? __filbuf(p) :
                        (int*)((p)->_ptr++))
#define putc_unlocked(x, p) (-(p)->_cnt < 0 ? __fsbuf(x, p) :
                          (int*)((p)->_ptr++ = (x)))
#define getc_unlocked() getc_unlocked(stdin)
#define putc_unlocked(x) putc_unlocked((x), stdout)

#define getc(p) (-(p)->_cnt < 0 ? __filbuf(p) :
              (int*)((p)->_ptr++))
#define putc(x, p) (-(p)->_cnt < 0 ? __fsbuf(x, p) :
                 (int*)((p)->_ptr++ = (x)))
#define getc() getc(stdin)
#define putc(x) putc((x), stdout)

/* Reentrant versions available as functions only */

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

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

6-78 LIBRARIES
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>`**

```c
typedef struct {
    int  quot;
    int  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 0
#define RAND_MAX 32767

extern unsigned char __ctype[];
#define MB_CUR_MAX (int)__ctype[520]
```
#define SNDZERO       0x001
#define SNDPIPE       0x002

#define RNORM         0x000
#define MSGD          0x001
#define MSGN          0x002
#define RMODEMASK     0x003
#define RPROTDAT      0x004
#define RPROTDIS      0x008
#define RPROTNORM     0x010

#define FLUSHR        0x01
#define FLUSHW        0x02
#define FLUSHRW       0x03
#define FLUSHBAND     0x04
#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)
#define ANYMARK 0x01
#define LASTMARK 0x02
Figure 6-96: `<stropts.h>`, Part 3 of 6

<table>
<thead>
<tr>
<th>Define</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#define STR 'S'&lt;&lt;8</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_NREAD (STR 01)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_PUSH (STR 02)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_POP (STR 03)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_LOOK (STR 04)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_FLUSH (STR 05)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_SRDOPT (STR 06)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_GRDOPT (STR 07)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_STR (STR 010)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_SETSIG (STR 011)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_GETSIG (STR 012)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_FIND (STR 013)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_LINK (STR 014)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_UNLINK (STR 015)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_PEEK (STR 017)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_FDINSERT (STR 020)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_SENDFD (STR 021)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_RECVFD (STR 016)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_SWROPT (STR 023)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_GWROPT (STR 024)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_LIST (STR 025)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_PLINK (STR 026)</code></td>
<td></td>
</tr>
<tr>
<td><code>#define I_PUNLINK (STR 027)</code></td>
<td></td>
</tr>
</tbody>
</table>
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_timeout;
    int  ic_len;
    char *ic_dp;
};

struct strbuf {
    int  maxlen;
    int  len;
    char *buf;
};

struct strpeek {
    struct strbuf    ctllbuf;
    struct strbuf    databuf;
    long              flags;
};

struct strfdinsert {
    struct strbuf    ctllbuf;
    struct strbuf    databuf;
    long              flags;
    int               fildes;
    int               offset;
};
```
Figure 6-99: `<stropts.h>`, Part 6 of 6

```c
struct strrecvfd {
    int fd;
    uid_t uid;
    gid_t gid;
    char fill[8];
};

#define FMAMESZ 8

struct str_mlist {
    char l_name[FMAMESZ+1];
};

struct str_list {
    int sl_nmods;
    struct str_mlist *sl_modlist;
};

struct bandinfo {
    unsigned char bi_pri;
    int bi_flag;
};
```
Figure 6-100: `<synch.h>*`, Part 1 of 3

```c
#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;
```
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 lwpc_cond;
    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**

```c
typedef volatile struct {
    mutex_t  rm_mutex;
    pid_t    rm_pid;
    thread_t rm_owner;
    int      rm_depth;
    long     filler;
} rmutex_t;

typedef volatile struct barrier 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>**

```c
#define SI86FPHW  40
#define FP_NO    0
#define FP_SW    1
#define FP_HW    2
#define FP_287   2
#define FP_387   3
```
```c
#define _POSIX_VDISABLE 0
#define CTRL(c) ((c)&037)
#define IBSHIFT 16
#define NCC 8
#define NCCS 19

typedef unsigned long tcflag_t;
typedef unsigned char cc_t;
typedef unsigned long speed_t;

struct termios {
    tcflag_t   c_iflag;
    tcflag_t   c_oflag;
    tcflag_t   c_cflag;
    tcflag_t   c_lflag;
    cc_t       c_cc[NCCS];
};
```
```c
#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 VSWITCH 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
```
# define CNUL 0
# define CDEL 0177
# define CESC ‘\\’
# define CINTR 0177
# define CQUIT 034
# define CERASE ‘#’
# define CKILL ‘@’
# define CEOT 04
# define CEOL 0
# define CEOL2 0
# define CEOF 04
# define CSTART 021
# define CSTOP 023
# define CSWITCH 032
# define CN SWITCH 0
# define CSUSP CTRL(‘z’)
# define CDSUSP CTRL(‘y’)
# define CRPRNT CTRL(‘x’)
# define CFLUSH CTRL(‘o’)
# define CWERASE CTRL(‘w’)
# define CLNEXT CTRL(‘v’)

Figure 6-106: <termios.h>, Part 3 of 10
Figure 6-107: <termios.h>, Part 4 of 10

```c
#define IGNBRK 0000001
#define BRKINT 0000002
#define IGNPAR 0000004
#define PARMRK 0000010
#define INPCK 0000020
#define ISTRIP 0000040
#define INLCR 0001000
#define IGNCR 0002000
#define ICRNL 0005000
#define IUCNL 0010000
#define IXON 0002000
#define IXANY 0004000
#define IXOFF 0010000
#define IMAXBEL 0020000
#define DOSMODE 0100000
```
# define OPOST 0000001
# define OLCUC 0000002
# define ONLKR 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 0010000
# define CR2 0020000
# define CR3 0030000
# 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
```
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 PAREN 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIG</td>
<td>0000001</td>
</tr>
<tr>
<td>ICANON</td>
<td>0000002</td>
</tr>
<tr>
<td>XCASE</td>
<td>0000004</td>
</tr>
<tr>
<td>ECHO</td>
<td>0000010</td>
</tr>
<tr>
<td>ECHOE</td>
<td>0000020</td>
</tr>
<tr>
<td>ECHOK</td>
<td>0000040</td>
</tr>
<tr>
<td>ECHONL</td>
<td>0000100</td>
</tr>
<tr>
<td>NOFLSH</td>
<td>0000200</td>
</tr>
<tr>
<td>TOSTOP</td>
<td>0000400</td>
</tr>
<tr>
<td>ECHOCTL</td>
<td>0001000</td>
</tr>
<tr>
<td>ECHOPRT</td>
<td>0002000</td>
</tr>
<tr>
<td>ECHOKE</td>
<td>0004000</td>
</tr>
<tr>
<td>DEFECCHO</td>
<td>0010000</td>
</tr>
<tr>
<td>FLUSHO</td>
<td>0020000</td>
</tr>
<tr>
<td>PENDIN</td>
<td>0040000</td>
</tr>
<tr>
<td>IEXTEN</td>
<td>0100000</td>
</tr>
</tbody>
</table>
### 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)

#define TIOCGWINSZ (TIOC|104)
#define TIOC SWINSZ (TIOC|103)

#define TCGETS (TIOC|13)
#define TCSETS (TIOC|14)
#define TCSANOW ('T'<<8|14)
#define TCSETSW (TIOC|15)
#define TCSADR AIN ('T'<<8|15)
#define TCSETSF (TIOC|16)
#define TCSAFLUSH ('T'<<8|16)
```
# define TCI_FLUSH 0
# define TCO_FLUSH 1
# define TCIO_FLUSH 2

# define TCOFF 0
# define TC0N 1
# define TCIOFF 2
# define TCI0N 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
# define B1200 9
# 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;
};
Figure 6-114: `<thread.h>*`, Part 1 of 2

```c
#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;

#define POLICY_PARAM_SZ PC_CLPARMSZ
```
**Figure 6-115:** `<thread.h>`, Part 2 of 2

```c
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>`

```c
#define TCL_BADADDR    1
#define TCL_BADOPT     2
#define TCL_NOPEER     3
#define TCL_PEERBADSTATE 4
#define TCL_DEFAULTADDRZ 4
```
### Figure 6-117: `<sys/ticots.h>`

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

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

```c
#define TCOO_NOPEER 1
#define TCOO_PEERNOROOMONQ 2
#define TCOO_PEERBADSTATE 3
#define TCOO_PEERINITIATED 4
#define TCOO_PROVIDERINITIATED 5
#define TCOO_DEFAULTADDRSZ 4
```

**NOTE**
The `sys/thdr.h` and `sys/timod.h` headers previously included in this document were unnecessary as they did not contain user level information and have therefore been removed from this document.
Figure 6-119: `<time.h>`

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

#define CLOCK_S_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;
Figure 6-121: <sys/times.h>

```
struct tms {
    clock_t tms_utime;
    clock_t tms_stime;
    clock_t tms_cutime;
    clock_t tms_cstime;
};
```

**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.

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.
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 TYSERR 8
#define TLOOK 9
#define TBADDATA 10
#define TBUFOVFLW 11
#define TFLOW 12
#define TNODATA 13
#define TNODIS 14
#define TNOUDERR 15
#define TBADFLAG 16
#define TNOREL 17
#define TNOTSUPPORT 18
#define TSTATECHNG 19
#define TNOSTRUCTYPE 20
#define TBADNAME 21
#define TBADQLEN 22
#define TADDRBUSY 23
#define TINDOUT 24
#define TPROVMISMATCH 25
#define TRESQLEN 26
#define TRESADDR 27
#define TQFULL 28
#define TPROTO 29
```
Figure 6-123: `<tiuser.h>`, Event Bitmasks

```
#define T_LISTEN 0x0001
#define T_CONNECT 0x0002
#define T_DATA 0x0004
#define T_EXDATA 0x0008
#define T_DISCONNECT 0x0010
#define T_ERROR 0x0020
#define T_UDERR 0x0040
#define T_ORDREL 0x0080
#define T_GODATA 0x0100
#define T_GOEXDATA 0x0200
#define T_EVENTS 0x03ff
```

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

```
#define T_MORE 0x0001
#define T_EXPEDITED 0x0002
#define T_NEGOTIATE 0x0004
#define T_CHECK 0x0008
#define T_DEFAULT 0x0010
#define T_SUCCESS 0x0020
#define T_FAILURE 0x0040
```
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_info` structure

```
#define T_SENDZERO 0x00000001
```
struct t_info {
    long addr;
    long options;
    long tsdu;
    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;
};

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 versions of t_open or t_getinfo will see the t_info structure with the flags member.
Figure 6-128: `<tiuser.h>`, Transport Interface Data Structures, 2 of 2

```c
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

```c
#define T_BIND 1
#define T_OPTMGMT 2
#define T_CALL 3
#define T_DIS 4
#define T_UNITDATA 5
#define T_UDERROR 6
#define T_INFO 7
```

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

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

**NOTE**

Differences between XTI and TLI have forced the value of T_ALL to change. The previous edition's T_ALL value will not produce the same results as the new T_ALL.
Figure 6-131: `<tiuser.h>`, Transport Interface States

```c
#define T_UNINIT  0
#define T_UNBND   1
#define T_IDLE    2
#define T_OUTCON  3
#define T_INCON   4
#define T_DATAFER 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

```c
#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
```
```c
typedef struct tsparms {
    short ts_uprllim;
    short ts_upri;
} tsparms_t;

typedef struct tsinfo {
    short ts_maxupri;
} tsinfo_t;

#define TS_NOCHANGE -32768
```
typedef unsigned char uchar_t;
typedef unsigned short ushort_t;
typedef unsigned int uint_t;
typedef unsigned long ulong_t;

typedef char * caddr_t;
typedef long daddr_t;
typedef long off_t;
typedef long id_t;
typedef int key_t;
typedef_ulong_t mode_t;
typedef long uid_t;
typedef uid_t gid_t;
typedef ulong_t nlink_t;
typedef ulong_t dev_t;
typedef ulong_t ino_t;
typedef long pid_t;
typedef uint_t size_t;
typedef long time_t;
typedef long clock_t;

typedef unsigned short ushort;
typedef unsigned long ulong;

typedef unsigned char u_char;
typedef unsigned short u_short;
typedef unsigned int u_int;
typedef unsigned long u_long;
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
typedef struct fpregs {
   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;
      int f_fpregs[62];
   } fp_reg_set;
   long f_wregs[33];
} fpregs_t;

typedef struct {
   gregset_t gregs;
   fpregs_t fpregs;
} mcontext_t;

typedef struct ucontext {
   u_long uc_flags;
   struct ucontext *uc_link;
   sigset_t uc_sigmask;
   stack_t uc_stack;
   mcontext_t uc_mcontext;
   long uc_filler[5];
} ucontext_t;
### Figure 6-137: `<sys/uio.h>`

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

### Figure 6-138: `<ulimit.h>`

```c
#define UL_GETFSIZE   1
#define UL_SETFSIZE   2
```

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

```c
#define R_OK           4
#define W_OK           2
#define X_OK           1
#define F_OK           0

#define F_UNLOCK       0
#define F_LOCK         1
#define F_TLOCK        2
#define F_TEST         3

#define SEEK_SET       0
#define SEEK_CUR       1
#define SEEK_END       2

#define _SC_ARG_MAX    1
#define _SC_CHILD_MAX  2
#define _SC_CLK_TCK    3
```
```
# 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 M
# define __CS_HOSTNAME 2 M
# define __CS_RELEASE 3 M
# define __CS_VERSION 4 M
# define __CS_MACHINE 5 M
# define __CS_ARCHITECTURE 6 M
# define __CS_HW_SERIAL 7 M
# define __CS_HW_PROVIDER 8 M
# define __CS_SRPC_DOMAIN 9 M
# define __CS_SYSNAME 11 M
```
Figure 6-140: `<unistd.h>`, Part 2 of 2

```c
#define _PC_LINK_MAX 1
#define _PC_MAX_CANON 2
#define _PC_MAX_INPUT 3
#define _PC_NAME_MAX 4
#define _PC_PATH_MAX 5
#define _PC_PIPE_BUF 6
#define _PC_NO_TRUNC 7
#define _PC_VDISABLE 8
#define _PC_CHOWN_RESTRICTED 9
#define _POSIX_JOB_CONTROL 1
#define _POSIX_SAVED_IDS 1
#define _POSIX_VDISABLE 0
#define _POSIX_VERSION *
#define _XOPEN_VERSION *

/* starred values vary and should be
   retrieved using sysconf() or pathconf() */
#define STDIN_FILENO 0
#define STDOUT_FILENO 1
#define STDERR_FILENO 2
```

Figure 6-141: `<utime.h>`

```c
struct utimbuf {
    time_t actime;
    time_t modtime;
};
```
#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];
};
Figure 6-143: <wait.h>

```c
#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 WEXITED(stat)  (WLOBYTE(stat)==0)
#define WIFSIGNALED(stat)  (WLOBYTE(stat)>0&&WHIBYTE(stat)==0)
#define WIFSTOPPED(stat)  (WLOBYTE(stat)==WSTOPFLG&&WHIBYTE(stat)!=0)
#define WIFCONTINUED(stat)  (WWORD(stat)==WCONTFLG)
#define WEXITSTATUS(stat)  WHIBYTE(stat)
#define WTERMSIG(stat)  (WLOBYTE(stat)&WSIGMASK)
#define WSTOPSIG(stat)  WHIBYTE(stat)
#define WCOREDUMP(stat)  ((stat)&WCOREFLG)
```
typedef long wchar_t;
typedef unsigned int size_t;
typedef long wint_t;

typedef struct
{
    wchar_t ;
    wchar_t ;
} mbstate_t;

#define NULL 0
#define WEOF (-1)

#define WCHAR_MAX 2147483647
#define WCHAR_MIN (-2147483647-1)

#define mbrlen(x, n, p) mbtowc((wchar_t *)0, x, n, p)
Figure 6-145: `<wctype.h>`, Part 1 of 3

```c
typedef long wint_t;
typedef unsigned 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 _E2 0x00000200
#define _E3 0x00000400
#define _E4 0x00000800
#define _E5 0x00001000
#define _E6 0x00002000
#define _E7 0x00004000
#define _E8 0x00008000
#define _E9 0x00010000
#define _E10 0x00020000
#define _E11 0x00040000
#define _E12 0x00080000
#define _E13 0x00100000
#define _E14 0x00200000
#define _E15 0x00400000
#define _E16 0x00800000
#define _E17 0x01000000
#define _E18 0x02000000
#define _E19 0x04000000
#define _E20 0x08000000
#define _E21 0x10000000
```
```c
#define _PD_ALNUM (_U | _L | _N)
#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)
#define iswalpha(c) __isw(c, _PD_ALPHA)
#define iswcntrl(c) __isw(c, _PD_CNTRL)
#define iswdigit(c) __isw(c, _PD_DIGIT)
#define iswgraph(c) __isw(c, _PD_GRAPH)
#define iswlower(c) __isw(c, _PD_LOWER)
#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)
#define towupper(c) __tow(c, _PD_LOWER)
#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)
```
Figure 6-147: `<wctype.h>`, Part 3 of 3

```c
#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 (__towctype(c, t));
    if ((1 + __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.
Figure 6-148: `<wordexp.h>`

```c
#define WRDE_APPEND 0001  M
#define WRDE_DOOFFS 0002  M
#define WRDE_NOCMD  0004  M
#define WRDE_REUSE   0010  M
#define WRDE_SHOKERR 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
{
    size_t we_wordc;  M
    char **we_wordv; M
    size_t we_offs;  M
} wordexp_t; M
```
X Window Data Definitions

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).
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
#define XA_PIXMAP   (Atom) 20
#define XA_POINT    (Atom) 21
#define XA_RECTANGLE (Atom) 22
#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
#define XA_RGB_RED_MAP   (Atom) 30
#define XA_STRING    (Atom) 31
#define XA_VISUALID  (Atom) 32
# 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)
Figure 6-152: `<X11/Composite.h>`

```c
extern WidgetClass compositeWidgetClass;
```

Figure 6-153: `<X11/Constraint.h>`

```c
extern WidgetClass constraintWidgetClass;
```

Figure 6-154: `<X11/Core.h>`

```c
extern WidgetClass coreWidgetClass;
```
Figure 6-155: `<X11/cursorfont.h>`, Part 1 of 3

```c
#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  
#define XC_circle 24  
#define XC_clock 26  
#define XC_coffee_mug 28  
#define XC_cross 30  
#define XC_cross_reverse 32  
#define XC_crosshair 34  
#define XC_diamond_cross 36  
#define XC_dot 38  
#define XC_dotbox 40  
#define XC_double_arrow 42  
#define XC_draft_large 44  
#define XC_draft_small 46  
#define XC_draped_box 48  
#define XC_exchange 50  
#define XC_fleur 52  
#define XC_gobbler 54  
#define XC_gumby 56  
#define XC_hand1 58  
#define XC_hand2 60
```
Figure 6-156: <X11/cursorfont.h>, Part 2 of 3

#define XC_heart 62
#define XC_icon 64
#define XC_iron_cross 66
#define XC_left_ptr 68
#define XC_left_side 70
#define XC_left_tee 72
#define XC_leftbutton 74
#define XC_ll_angle 76
#define XC_lr_angle 78
#define XC_man 80
#define XC_middlebutton 82
#define XC_mouse 84
#define XC_pencil 86
#define XC_pirate 88
#define XC_plus 90
#define XC_question_arrow 92
#define XC_right_ptr 94
#define XC_right_side 96
#define XC_right_tee 98
#define XC_rightbutton 100
#define XC_rtl_logo 102
#define XC_sailboat 104
#define XC_sb_down_arrow 106
#define XC_sb_h_double_arrow 108
#define XC_sb_left_arrow 110
#define XC_sb_right_arrow 112
#define XC_sb_up_arrow 114
#define XC_sb_v_double_arrow 116
#define XC_shuttle 118
#define XC_sizing 120
#define XC_spider 122
#define XC_spraycan 124
### Figure 6-157: `<X11/cursorfont.h>`, Part 3 of 3

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>XC_star</code></td>
<td>126</td>
</tr>
<tr>
<td><code>XC_target</code></td>
<td>128</td>
</tr>
<tr>
<td><code>XC_tcross</code></td>
<td>130</td>
</tr>
<tr>
<td><code>XC_top_left_arrow</code></td>
<td>132</td>
</tr>
<tr>
<td><code>XC_top_left_corner</code></td>
<td>134</td>
</tr>
<tr>
<td><code>XC_top_right_corner</code></td>
<td>136</td>
</tr>
<tr>
<td><code>XC_top_side</code></td>
<td>138</td>
</tr>
<tr>
<td><code>XC_top_tee</code></td>
<td>140</td>
</tr>
<tr>
<td><code>XC_trek</code></td>
<td>142</td>
</tr>
<tr>
<td><code>XC_ul_angle</code></td>
<td>144</td>
</tr>
<tr>
<td><code>XC_umbrella</code></td>
<td>146</td>
</tr>
<tr>
<td><code>XC_ur_angle</code></td>
<td>148</td>
</tr>
<tr>
<td><code>XC_watch</code></td>
<td>150</td>
</tr>
<tr>
<td><code>XC_xterm</code></td>
<td>152</td>
</tr>
</tbody>
</table>
typedef char *String;

#define XtNumber(arr)\n     ((Cardinal) (sizeof(arr) / sizeof(arr[0])))

typedef void **Widget;
typedef Widget **WidgetList;
typedef void *CompositeWidget;
typedef void *WidgetClass;
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 XtCacheAll 0x002
#define XtCacheByDisplay 0x003
#define XtCacheRefCount 0x100

typedef char Boolean;
typedef long XrArgVal;
typedef unsigned char XrEnum;

typedef unsigned int Cardinal;
typedef unsigned short Dimension;
typedef short Position;

typedef void *XtPointer;
typedef void *XtTranslations;
typedef void *XtAccelerators;
typedef unsigned int Modifiers;

#define XtCWQueryOnly (1 << 7)
#define XtSMDontChange 5

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;
```c
#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;
```
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;
    XtPointer default_addr;
} XtResource, *XtResourceList;

typedef struct {
    char match;
    String substitution;
} SubstitutionRec, *Substitution;

typedef Boolean (*XtFilePredicate);
typedef XtPointer XtRequestId;

extern XtConvertArgRec const colorConvertArgs[];
extern XtConvertArgRec const screenConvertArg[];
#define XtAllEvents ((EventMask) -1L)
#define XtIMXEvent 1
#define XtIMTimer 2
#define XtIMAAlternateInput 4
#define XtIMAAll (XtIMXEvent | XtIMTimer | XtIMAAlternateInput)

#define XtOffsetOf(s_type,field) XtOffset(s_type*,field)
#define XtNew(type) ((type *) XtMalloc((unsigned) sizeof(type))
#define XT_CONVERT_FAIL (Atom)0x80000001

#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, (WidgClass)overrideShellWidgClass, \  (WidgClass)shellWidgClass, (XtEnum)0x20))
#define XtIsWMShell(widget) \  (_XtCheckSubclassFlag(widget, (XtEnum)0x40))
#define XtIsVendorShell(widget) \  (_XtIsSubclassOf(widget, (WidgClass)vendorShellWidgClass, \  (WidgClass)wmShellWidgClass, (XtEnum)0x40))
#define XtIsTransientShell(widget) \  (_XtIsSubclassOf(widget, (WidgClass)transientShellWidgClass, \  (WidgClass)wmShellWidgClass, (XtEnum)0x40))
#define XtIsTopLevelShell(widget) \  (_XtCheckSubclassFlag(widget, (XtEnum)0x80))
#define XtIsApplicationShell(widget) \  (_XtIsSubclassOf(widget, (WidgClass)applicationShellWidgClass, \  (WidgClass)topLevelShellWidgClass, (XtEnum)0x80))
Figure 6-163: `<X11/Intrinsic.h>, Part 6 of 6`

```c
#define XtSetArg(x, y, z)
    ((void) ( (x).name = (y), (x).value = (int)z ))
#define XtOffset(x, y)
    ((Cardinal) (((char *)(x)) - (char *)NULL))
#define XtVaNestedList "XtVaNestedList"
#define XtVaTypedArg "XtVaTypedArg"
#define XtUnspecifiedPixmap ((Pixmap)2)
#define XtUnspecifiedShellInt 0
#define XtUnspecifiedWindow 0
#define XtDefaultForeground "XtDefaultForeground"
#define XtDefaultBackground "XtDefaultBackground"
#define XtDefaultFont "XtDefaultFont"
#define XtDefaultFontSet "XtDefaultFontSet"
```

---

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

```c
extern WidgetClass objectClass;
```

---

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

```c
extern WidgetClass rectObjClass;
```
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

#define ShapeNotifyMask (1L << 0)
#define ShapeNotify    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;
```
typedef unsigned long XID;
typedef XID Window;
typedef XID Drawable;
typedef XID Font;
typedef XIDPixmap;
typedef XID Cursor;
typedef XIDColormap;
typedef XIDGContext;
typedef XIDKeySym;
typedef unsigned long Atom;
typedef unsigned long VisualID;
typedef unsigned long Time;
typedef unsigned char KeyCode;

#define AllTemporary 0L
#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 1L
#define PointerWindow 0L
#define PointerRoot 1L
Figure 6-170: <X11/X.h>, Part 2 of 12

```c
#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 SubstructureRedirectMask(1L<<20)
#define FocusChangeMask  (1L<<21)
#define PropertyChangeMask(1L<<22)
#define ColormapChangeMask(1L<<23)
#define OwnerGrabButtonMask(1L<<24)
```
# 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
# define MapRequest 20
# define ReparentNotify 21
# define ConfigureNotify 22
# define ConfigureRequest 23
# define GravityNotify 24
# define ResizeRequest 25
# define CirculateNotify 26
# define CirculateRequest 27
# define PropertyNotify 28
# define SelectionClear 29
# define SelectionRequest 30
# define SelectionNotify 31
# define ColormapNotify 32
# define ClientMessage 33
# define MappingNotify 34
# define LASTEvent 35

/* must be bigger than any event */
```c
#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)
#define Button2Mask  (1<<9)
#define Button3Mask  (1<<10)
#define Button4Mask  (1<<11)
#define Button5Mask  (1<<12)
#define AnyModifier (1<<15)
#define Button1      1
#define Button2      2
#define Button3      3
#define Button4      4
#define Button5      5
#define NotifyNormal 0
#define NotifyGrab   1
#define NotifyUngrab 2
#define NotifyWhileGrabbed 3
#define NotifyHint   1
#define NotifyAncestor 0
#define NotifyVirtual 1
#define NotifyInferior 2
#define NotifyNonlinear 3
#define NotifyNonlinearVirtual 4
#define NotifyPointer 5
#define NotifyPointerRoot 6
#define NotifyDetailNone 7
```
# define VisibilityUnobscured 0
# define VisibilityPartiallyObscured 1
# define VisibilityFullyObscured 2

# define PlaceOnTop 0
# define PlaceOnBottom 1

# define PropertyNewValue 0
# define PropertyDelete 1

# define ColormapUninstalled 0
# define ColormapInstalled 1

# define GrabModeSync 0
# define GrabModeAsync 1

# define GrabSuccess 0
# define AlreadyGrabbed 1
# define GrabInvalidTime 2
# define GrabNotViewable 3
# define GrabFrozen 4

# define AsyncPointer 0
# define SyncPointer 1
# define ReplayPointer 2
# define AsyncKeyboard 3
# define SyncKeyboard 4
# define ReplayKeyboard 5
# define AsyncBoth 6
# define SyncBoth 7

# define RevertToNone (int)None
# define RevertToPointerRoot (int)PointerRoot
# define RevertToParent 2
Figure 6-174: `<X11/X.h>`, Part 6 of 12

```c
#define Success 0
#define BadRequest 1
#define BadValue 2
#define BadWindow 3
#define BadPixmap 4
#define BadAtom 5
#define BadCursor 6
#define BadFont 7
#define BadMatch 8
#define BadDrawable 9
#define BadAccess 10
#define BadAlloc 11
#define BadColor 12
#define BadGC 13
#define BadIDChoice 14
#define BadName 15
#define BadLength 16
#define BadImplementation 17

#define InputOutput 1
#define InputOnly 2

#define CWBackPixmap (1L<<0)
#define CWBackPixel (1L<<1)
#define CWBorderPixmap (1L<<2)
#define CWBorderPixel (1L<<3)
#define CBitGravity (1L<<4)
#define CWinGravity (1L<<5)
#define CWBackingStore (1L<<6)
#define CWBackingPlanes (1L<<7)
#define CWBackingPixel (1L<<8)
#define CWOverrideRedirect (1L<<9)
#define CWSaveUnder (1L<<10)
#define CWEventMask (1L<<11)
#define CWDon'tPropagate (1L<<12)
#define CWColormap (1L<<13)
#define CWCursor (1L<<14)
```

System Data Interfaces 6-147
Figure 6-175: `<X11/X.h>`, Part 7 of 12

```c
#define CWX (1<<0)
#define CWY (1<<1)
#define CWWidth (1<<2)
#define CWHHeight (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
#define CenterGravity 5
#define EastGravity 6
#define SouthWestGravity 7
#define SouthGravity 8
#define SouthEastGravity 9
#define StaticGravity 10
#define UnmapGravity 0
#define NotUseful 0
#define WhenMapped 1
#define Always 2
#define IsUnmapped 0
#define IsUnviewable 1
#define IsViewable 2
#define SetModeInsert 0
#define SetModeDelete 1
#define DestroyAll 0
#define RetainPermanent 1
#define RetainTemporary 2
```
Figure 6-176: <X11/X.h>, Part 8 of 12

```c
#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

#define GXclear       0x0
#define GXand         0x1
#define GXandReverse  0x2
#define GXcopy        0x3
#define GXandInverted 0x4
#define GXnoop        0x5
#define GXxor         0x6
#define GXor          0x7
#define GXnor         0x8
#define GXequiv       0x9
#define GXinvert      0xa
#define GXorReverse   0xb
#define GXcopyInverted 0xc
#define GXorInverted  0xd
#define GXnand        0xe
#define GXset         0xf

#define LineSolid    0
#define LineOnOffDash 1
#define LineDoubleDash 2
#define CapNotLast   0
#define CapButt      1
#define CapRound     2
#define CapProjecting 3
```
Figure 6-177: <X11/X.h>, Part 9 of 12

```c
#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

#define Complex 0
#define Nonconvex 1
#define Convex 2

#define ArcChord 0
#define ArcPieSlice 1
```
System Data Interfaces

6-151
# define CursorShape 0
# define TileShape 1
# define StippleShape 2

# define AutoRepeatModeOff 0
# define AutoRepeatModeOn 1
# define AutoRepeatModeDefault 2

# define LedModeOff 0
# define LedModeOn 1

#define KBKeyClickPercent 1L<<0
#define KBBellPercent 1L<<1
#define KBBellPitch 1L<<2
#define KBBellDuration 1L<<3
#define KBLed 1L<<4
#define KBLedMode 1L<<5
#define KBKey 1L<<6
#define KBAutoRepeatMode 1L<<7

#define MappingSuccess 0
#define MappingBusy 1
#define MappingFailed 2

#define MappingModifier 0
#define MappingKeyboard 1
#define MappingPointer 2
#define DontPreferBlanking 0
#define PreferBlanking 1
#define DefaultBlanking 2

#define DontAllowExposures 0
#define AllowExposures 1
#define DefaultExposures 2
Figure 6-180: `<X11/X.h>`, Part 12 of 12

```c
#define ScreenSaverReset 0
#define ScreenSaverActive 1

#define EnableAccess 1
#define DisableAccess 0
#define StaticGray 0
#define GrayScale 1

#define StaticColor 2
#define PseudoColor 3
#define TrueColor 4
#define DirectColor 5

#define LSBFirst 0
#define MSBFirst 1
```
# define XcmsFailure 0
# define XcmsSuccess 1
# define XcmsSuccessWithCompression 2

# define XcmsUndefinedFormat (XcmsColorFormat)0x00000000
# define XcmsCIELuVYFormat (XcmsColorFormat)0x00000001
# define XcmsCIELabFormat (XcmsColorFormat)0x00000002
# define XcmsCIELuvFormat (XcmsColorFormat)0x00000003
# define XcmsTekHVFormat (XcmsColorFormat)0x00000004
# define XcmsRGBFormat (XcmsColorFormat)0x00000005
# define XcmsRGBiFormat (XcmsColorFormat)0x00000006

# 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;
```c
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;
} XcmsCIExyX;

typedef struct {
    XcmsFloat L_star;
    XcmsFloat a_star;
    XcmsFloat b_star;
} XcmsCIELab;
```
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;
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];
} XcmsPerScrntnfo;

typedef void *XcmsCCC;

typedef Status (*XcmsConversionProc)();
typedef XcmsConversionProc *XcmsFuncListPtr;
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;
typedef void *XPointer;

#define Bool int
#define Status int
#define True 1
#define False 0
#define QueuedAlready 0
#define QueuedAfterReading 1
#define QueuedAfterFlush 2
#define AllPlanes ((unsigned long)~0L)

typedef void XExtData;
typedef void XExtCodes;
typedef struct {
    int depth;
    int bits_per_pixel;
    int scanline_pad;
} XPixmapFormatValues;
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;
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;
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;
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;
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;
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;
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;
```c
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;
```
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;
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;

System Data Interfaces 6-169
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;
```c
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;
```
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;
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;
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;
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 l[5];
    } data;
} XClientMessageEvent;
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;
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;
    XResizeRequestEvent xresizerequest;
    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;
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;
    Bool all_chars_exist;
    unsigned default_char;
    int n_properties;
    XFontProp *properties;
    XCharStruct min_bounds;
    XCharStruct max_bounds;
    XCharStruct *per_char;
    int ascent;
    int descent;
} XFontStruct;
typedef struct {
    char *chars;
    int nchars;
    int delta;
    Font font;
} XTextItem;

typedef struct {
    unsigned char byte1;
    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;
} XDataObject;

typedef struct {
    XRectangle max_ink_extent;
    XRectangle max_logical_extent;
} XFontSetExtents;

typedef struct _dummy XFontSet;
```c
typedef struct {
    char   *chars;
    int    nchars;
    int    delta;
    XFontSet *font_set;
} XmbTextItem;

typedef struct {
    wchar_t *chars;
    int    nchars;
    int    delta;
    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 XIMPredictArea    0x0001L
#define XIMPredictCallbacks 0x0002L
#define XIMPredictPosition 0x0004L
#define XIMPredictNothing  0x0008L
#define XIMPredictNone     0x0010L
#define XIMStatusArea      0x0100L
#define XIMStatusCallbacks 0x0200L
#define XIMStatusNothing   0x0400L
#define XIMStatusNone      0x0800L
```
#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 XNFreeeditStartCallback "preeditStartCallback"
#define XNFreeeditDoneCallback "preeditDoneCallback"
#define XNFreeeditDrawCallback "preeditDrawCallback"
#define XNFreeeditCaretCallback "preeditCaretCallback"
#define XNFreeeditAttributes "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"


```c
#define XBufferOverflow  -1
#define XLookupNone      1
#define XLookupChars     2
#define XLookupKeySym    3
#define XLookupBoth      4

typedef XPointer XVaNestedList;

typedef struct {
    XPointer client_data;
    XIMProc callback;
} XIMProc;

typedef unsigned long XIMFeedback;

#define XIMReverse      1
#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;
```
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;
```c
typedef enum {
    XIMTextType,
    XIMBitmapType
} XIMStatusDataType;

typedef struct _XIMStatusDrawCallbackStruct {
    XIMStatusDataType type;
    union {
        XIMText *text;
       Pixmap bitmap;
    } data;
} XIMStatusDrawCallbackStruct;
```
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
typedef enum {
    XrmoptionNoArg,
    XrmoptionIsArg,
    XrmoptionStickyArg,
    XrmoptionSepArg,
    XrmoptionResArg,
    XrmoptionSkipArg,
    XrmoptionSkipLine,
    XrmoptionSkipNArgs
} XrmOptionKind;

typedef struct {
    char *option;
    char *specifier;
    XrmOptionKind argKind;
    XPointer value;
} XrmOptionDescRec, *XrmOptionDescList;
Figure 6-215: \(<X11/Xutil.h>\), Part 1 of 5

```c
#define NoValue 0x0000
#define XValue 0x0001
#define YValue 0x0002
#define WidthValue 0x0004
#define HeightValue 0x0008
#define AllValues 0x000F
#define XNegative 0x0010
#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)
#define PPosition (1L << 2)
#define PSize (1L << 3)
#define PMinSize (1L << 4)
#define PMaxSize (1L << 5)
#define PResizeInc (1L << 6)
#define PAspect (1L << 7)
#define PBaseSize (1L << 8)
#define PWinGravity (1L << 9)
#define PAllHints (PPosition|PSize|PMinSize|PMaxSize|PResizeInc|PAspect)
```

System Data Interfaces 6-187
typedef struct {
    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)
#define StateHint (1L << 1)
#define IconPixmapHint (1L << 2)
#define IconWindowHint (1L << 3)
#define IconPositionHint (1L << 4)
#define IconMaskHint (1L << 5)
#define WindowGroupHint (1L << 6)
#define AllHints (InputHint|StateHint|
                  IconPixmapHint|IconWindowHint|
                  IconPositionHint|IconMaskHint|WindowGroupHint)

#define WithdrawnState 0
#define NormalState 1
#define IconicState 3

typedef struct {
    unsigned char *value;
    Atom encoding;
    int format;
    unsigned long nitems;
} XTextProperty;

#define XNoMemory -1
#define XLocaleNotSupported -2
#define XConverterNotFound -3
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;
#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;
    int screen;
    int depth;
    int class;
    unsigned long red_mask;
    unsigned long green_mask;
    unsigned long blue_mask;
    int colormap_size;
    int bits_per_rgb;
} XVisualInfo;
Figure 6-219: <X11/Xutil.h>, Part 5 of 5

```c
#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 VisualBitsPerRGBMask 0x100
#define VisualAllMask 0xFF

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 0
#define BitmapOpenFailed 1
#define BitmapFileInvalid 2
#define BitmapNoMemory 3
#define XCSUCCESS 0
#define XCNOMEM 1
#define XCNOMEM 2
#define XUniqueContext() ((XContext) XrmUniqueQuark())
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).
Figure 6-220: `<Xm/ArrowB.h>`*

typedef struct _XmArrowButtonClassRec * XmArrowButtonWidgetClass;
typedef struct _XmArrowButtonRec       * XmArrowButtonWidget;

---

Figure 6-221: `<Xm/ArrowBG.h>`*

typedef struct _XmArrowButtonGadgetClassRec * XmArrowButtonGadgetClass;
typedef struct _XmArrowButtonGadgetRec       * XmArrowButtonGadget;

---

Figure 6-222: `<Xm/BulletinB.h>`*

typedef struct _XmBulletinBoardClassRec * XmBulletinBoardWidgetClass;
typedef struct _XmBulletinBoardRec          * XmBulletinBoardWidget;

---

Figure 6-223: `<Xm/CascadeB.h>`*

typedef struct _XmCascadeButtonRec          * XmCascadeButtonWidget;
typedef struct _XmCascadeButtonClassRec     * XmCascadeButtonWidgetClass;
Figure 6-224: <Xm/CascadeBG.h>*

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

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

typedef struct _XmCommandClassRec * XmCommandWidgetClass;
typedef struct _XmCommandRec * XmCommandWidget;
Figure 6-226: <Xm/CutPaste.h>*

```c
#define XmClipboardFail 0
#define XmClipboardSuccess 1
#define XmClipboardTruncate 2
#define XmClipboardLocked 4
#define XmClipboardBadFormat 5
#define XmClipboardNoData 6
#define ClipboardFail 0

#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>*

```c
typedef struct _XmDialogShellClassRec * XmDialogShellWidgetClass;
typedef struct _XmDialogShellRec * XmDialogShellWidget;
```
Figure 6-228: `<Xm/Display.h>`

```c
enum {
    XmDRAG_NONE,
    XmDRAG_DROP_ONLY,
    XmDRAG_PREFER_PREREGISTER,
    XmDRAG_PREREGISTER,
    XmDRAG_PREFER_DYNAMIC,
    XmDRAG_DYNAMIC,
    XmDRAG_PREFER RECEIVER
};

typedef struct _XmDisplayRec *XmDisplay;
typedef struct _XmDisplayClassRec *XmDisplayClass;
```
Figure 6-229: <Xm/DragC.h">*, Part 1 of 4

```c
#define XmDROP_MOVE (1L << 0)
#define XmDROP_COPY (1L << 1)
#define XmDROP_LINK (1L << 2)

#define XmHELP 2
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,
      XmDROP_MOTION, XmDROP_SITE_ENTER,
      XmDROP_SITE_LEAVE, XmDROP_START,
      XmDROP_FINISH, XmDROP_FINISH,
      XmOPERATION_CHANGED
      };

enum { XmDROP, XmDROP_HELP,
      XmDROP_CANCEL, XmDROP_INTERRUPT
      };
```
# define XmDROP_NOOP 0L

enum{
    XmBLENDFL_ALL,       XmBLENDFL_STATE_SOURCE,
    XmBLENDFL_JUST_SOURCE,  XmBLENDFL_NONE
};

enum{
    XmDROP_FAILURE, XmDROP_SUCCESS
};

enum{
    XmCR_TOP_LEVEL_ENTER,     XmCR_TOP_LEVEL_LEAVE,
    XmCR_DRA_G_MOTION,       XmCR_DROP_SITE_ENTER,
    XmCR_DROP_SITE_LEAVE,    XmCR_DROP_START,
    XmCR_DROP_FINISH,        XmCR_DRA_G_DRO_P_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{
    int reason;
    XEvent *event;
    Time timeStamp;
    Screen *screen;
    Window window;
    Position x, y;
    unsigned char dragProtocolStyle;
    Atom iccHandle;
}XmTopLevelEnterCallbackStruct, *XmTopLevelEnterCallback;
typedef struct _XmTopLevelLeaveCallbackStruct{
    int reason;
    XEvent *event;
    Time timeStamp;
    Screen *screen;
    Window window;
} XmTopLevelLeaveCallbackStruct, *XmTopLevelLeaveCallback;

typedef struct _XmDropSiteEnterCallbackStruct{
    int reason;
    XEvent *event;
    Time timeStamp;
    unsigned char operation;
    unsigned char operations;
    unsigned char dropSiteStatus;
    Position x, y;
} XmDropSiteEnterCallbackStruct, *XmDropSiteEnterCallback;

typedef struct _XmDropSiteLeaveCallbackStruct{
    int 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;
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 icchHandle;
} 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;
Figure 6-233: <Xm/DragIcon.h>*

```c
enum {
    XmATTACH_NORTH_WEST,
    XmATTACH_NORTH,
    XmATTACH_NORTH_EAST,
    XmATTACH_EAST,
    XmATTACH_SOUTH_EAST,
    XmATTACH_SOUTH,
    XmATTACH_SOUTH_WEST,
    XmATTACH_WEST,
    XmATTACH_CENTER,
    XmATTACH_HOT
};

typedef struct _XmDragIconRec *XmDragIconObject;
typedef struct _XmDragIconClassRec *XmDragIconObjectClass;
```

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

```c
typedef struct _XmDragOverShellRec *XmDragOverShellWidget;
typedef struct _XmDragOverShellClassRec *XmDragOverShellWidgetClass;
```
Figure 6-235: `<Xm/DrawingA.h>`

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

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

typedef struct _XmDrawnButtonClassRec *XmDrawnButtonWidgetClass;
typef struct _XmDrawnButtonRec *XmDrawnButtonWidget;
```c
#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;
```
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;
    Pixmap topShadowPixmap;
    Pixel bottomShadowColor;
    Pixmap bottomShadowPixmap;
    Dimension shadowThickness;
    Pixel highlightColor;
    Pixmap highlightPixmap;
    Dimension highlightThickness;
    Dimension borderWidth;
} XmDropSiteVisualsRec, *XmDropSiteVisuals;

typedef struct _XmDropSiteManagerClassRec *XmDropSiteManagerObjectClass;
typedef struct _XmDropSiteManagerRec *XmDropSiteManagerObject;
Figure 6-239: <Xm/DropTrans.h>*

```c
#define XmTRANSFER_FINE 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>*

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

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

```c
typedef struct _XmFormClassRec * XmFormWidgetClass;
typedef struct _XmFormRec * XmFormWidget;
```
Figure 6-242: `<Xm/Frame.h>`

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

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

typedef struct _XmLabelClassRec * XmLabelWidgetClass;
typedef struct _XmLabelRec * XmLabelWidget;

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

typedef struct _XmLabelGadgetClassRec * XmLabelGadgetClass;
typedef struct _XmLabelGadgetRec * XmLabelGadget;
typedef struct _XmLabelGCacheObjRec * XmLabelGCacheObject;
Figure 6-245: <Xm/List.h>*

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

typedef struct _ XmListClassRec * XmListWidgetClass;
typedef struct _ XmListRec * XmListWidget;
```

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

```c
typedef struct _ XmMainWindowClassRec * XmMainWindowWidgetClass;
typedef struct _ XmMainWindowRec * XmMainWindowWidget;
```

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

```c
typedef struct _ XmMenuShellClassRec * XmMenuShellWidgetClass;
typedef struct _ XmMenuShellWidgetRec * XmMenuShellWidget;
```
typedef struct _XmMessageBoxClassRec * XmMessageBoxWidgetClass;
typedef struct _XmMessageBoxRec * XmMessageBoxWidget;
Figure 6-249: `<Mrm/MrmPublic.h>`, Part 1 of 3

```c
#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 MrmNULL_GROUP 6
#define MrmNULL_TYPE 8
#define MrmWRONG_GROUP 10
#define MrmWRONG_TYPE 12
#define MrmOUT_OF_RANGE 14
#define MrmBAD_RECORD 16
#define 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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define MrmNULL_ROUTINE</td>
<td>56</td>
</tr>
<tr>
<td>#define MrmVEC_TOO_BIG</td>
<td>58</td>
</tr>
<tr>
<td>#define MrmBAD_HIERARCHY</td>
<td>60</td>
</tr>
<tr>
<td>#define MrmBAD_CLASS_CODE</td>
<td>62</td>
</tr>
<tr>
<td>#define MrmDISPLAY_NOT_OPENED</td>
<td>63</td>
</tr>
<tr>
<td>#define MrmEOF</td>
<td>64</td>
</tr>
<tr>
<td>#define MrmUNRESOLVED_REFS</td>
<td>65</td>
</tr>
<tr>
<td>#define MrmNcreateCallback</td>
<td>&quot;createCallback&quot;</td>
</tr>
<tr>
<td>#define MrmCR_CREATE</td>
<td>XmCR_CREATE</td>
</tr>
<tr>
<td>#define MrmwcUnknown</td>
<td>1</td>
</tr>
<tr>
<td>#define MrmRtypeMin</td>
<td>1</td>
</tr>
<tr>
<td>#define MrmRtypeInteger</td>
<td>1</td>
</tr>
<tr>
<td>#define MrmRtypeBoolean</td>
<td>2</td>
</tr>
<tr>
<td>#define MrmRtypeChar8</td>
<td>3</td>
</tr>
<tr>
<td>#define MrmRtypeChar8Vector</td>
<td>4</td>
</tr>
<tr>
<td>#define MrmRtypeCString</td>
<td>5</td>
</tr>
<tr>
<td>#define MrmRtypeCStringVector</td>
<td>6</td>
</tr>
<tr>
<td>#define MrmRtypeFloat</td>
<td>7</td>
</tr>
<tr>
<td>#define MrmRtypeCallback</td>
<td>9</td>
</tr>
<tr>
<td>#define MrmRtypePixmapImage</td>
<td>10</td>
</tr>
<tr>
<td>#define MrmRtypePixmapDDIF</td>
<td>11</td>
</tr>
<tr>
<td>#define MrmRtypeResource</td>
<td>12</td>
</tr>
<tr>
<td>#define MrmRtypeNull</td>
<td>13</td>
</tr>
<tr>
<td>#define MrmRtypeAddrName</td>
<td>14</td>
</tr>
<tr>
<td>#define MrmRtypeIconImage</td>
<td>15</td>
</tr>
<tr>
<td>#define MrmRtypeFont</td>
<td>16</td>
</tr>
<tr>
<td>#define MrmRtypeFontList</td>
<td>17</td>
</tr>
<tr>
<td>#define MrmRtypeColor</td>
<td>18</td>
</tr>
<tr>
<td>#define MrmRtypeColorTable</td>
<td>19</td>
</tr>
<tr>
<td>#define MrmRtypeAny</td>
<td>20</td>
</tr>
<tr>
<td>#define MrmRtypeTransTable</td>
<td>21</td>
</tr>
<tr>
<td>#define MrmRtypeClassRecName</td>
<td>22</td>
</tr>
<tr>
<td>#define MrmRtypeIntegerVector</td>
<td>23</td>
</tr>
</tbody>
</table>
Figure 6-251: `<Mrm/MrmPublic.h>`, Part 3 of 3

```c
# define MrmRtypeXBitmapFile 24
# define MrmRtypeCountedVector 25
# define MrmRtypeKeysym 26
# define MrmRtypeSingleFloat 27
# 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 1
```
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)

#define MWM_FUNC_ALL (1L << 0)
#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_RESIZETH (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
typedef struct
{
   long flags;
   Window wm_window;
} MotifWmInfo;

typedef MotifWmInfo MwmInfo;

#define MWM_INFO_STARTUP_STANDARD   (1L << 0)
#define MWM_INFO_STARTUP_CUSTOM     (1L << 1)

typedef struct
{
   CARD32 flags;
   CARD32 functions;
   CARD32 decorations;
   INT32 inputMode;
   CARD32 status;
} PropMotifWmHints;

typedef PropMotifWmHints PropMwmHints;

#define PROP_MOTIF_WM_HINTS_ELEMENTS 5
#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

```c
#define _XA_MOTIF_WM_MENU "_MOTIF_WM_MENU"
#define _XA_WM_MENU _XA_MOTIF_WM_MENU

typedef struct {
    CARD32 flags;
    CARD32 wmWindow;
} PropMotifWmInfo;

typedef PropMotifWmInfo PropWmInfo;

#define PROP_MOTIF_WM_INFO_ELEMENTS 2
#define PROP_MWM_INFO_ELEMENTS PROP_MOTIF_WM_INFO_ELEMENTS

#define _XA_MOTIF_WM_INFO "_MOTIF_WM_INFO"
#define _XA_WM_INFO _XA_MOTIF_WM_INFO

#define _XA_MOTIF_BINDINGS "_MOTIF_BINDINGS"
```

Figure 6-255: `<Xm/PanedW.h>`

```c
typedef struct _XmPanedWindowClassRec *XmPanedWindowWidgetClass;
typedef struct _XmPanedWindowRec *XmPanedWindowWidget;
```
typedef struct _XmPushButtonClassRec *XmPushButtonWidgetClass;
typedef struct _XmPushButtonRec *XmPushButtonWidget;

typedef struct _XmPushButtonGadgetClassRec *XmPushButtonGadgetClass;
typedef struct _XmPushButtonGadgetRec *XmPushButtonGadget;
typedef struct _XmPushButtonGCacheObjRec *XmPushButtonGCacheObject;

#define XmREP_TYPE_INVALID 0xFFF

typedef unsigned short XmRepTypeId;
typedef struct
{
    String rep_type_name;
    String *value_names;
    unsigned char *values;
    unsigned char num_values;
    Boolean reverse_installed;
    XmRepTypeId rep_type_id;
}XmRepTypeEntryRec, *XmRepTypeEntry, XmRepTypeListRec, *XmRepTypeList;
Figure 6-259: `<Xm/RowColumn.h>`

typedef struct _XmRowColumnClassRec * XmRowColumnWidgetClass;
typedef struct _XmRowColumnRec * XmRowColumnWidget;

Figure 6-260: `<Xm/Scale.h>`

typedef struct _XmScaleClassRec * XmScaleWidgetClass;
typedef struct _XmScaleRec * XmScaleWidget;

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;
Figure 6-263: <Xm/ScrolledW.h>*

typedef struct _XmScrolledWindowClassRec * XmScrolledWindowWidgetClass;
typedef struct _XmScrolledWindowRec * XmScrolledWindowWidget;

Figure 6-264: <Xm/SelectionBox.h>*

typedef struct _XmSelectionBoxClassRec * XmSelectionBoxWidgetClass;
typedef struct _XmSelectionBoxRec * XmSelectionBoxWidget;

Figure 6-265: <Xm/SeparatorGadget.h>*

typedef struct _XmSeparatorGadgetClassRec * XmSeparatorGadgetClass;
typedef struct _XmSeparatorGadgetRec * XmSeparatorGadget;
typedef struct _XmSeparatorGCacheObjRec * XmSeparatorGCacheObject;

Figure 6-266: <Xm/Separator.h>*

typedef struct _XmSeparatorClassRec * XmSeparatorWidgetClass;
typedef struct _XmSeparatorRec * XmSeparatorWidget;

System Data Interfaces 6-217
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;
Figure 6-271: `<Xm/VendorS.h>`

typedef struct _XmVendorShellRec *XmVendorShellWidget;
typedef struct _XmVendorShellClassRec *XmVendorShellWidgetClass;

Figure 6-272: `<Xm/VirtKeys.h>` *, Part 1 of 2

```c
#define _OSF_Keysyms
#define osfXK_BackSpace 0x1004FF08
#define osfXK_Insert 0x1004FF63
#define osfXK_Delete 0x1004FFFF
#define osfXK_Copy 0x1004FF02
#define osfXK_Cut 0x1004FF03
#define osfXK_Paste 0x1004FF04
#define osfXK_AddMode 0x1004FF31
#define osfXK_PrimaryPaste 0x1004FF32
#define osfXK_QuickPaste 0x1004FF33
#define osfXK_PageLeft 0x1004FF40
#define osfXK_PageUp 0x1004FF41
#define osfXK_PageDown 0x1004FF42
#define osfXK_PageRight 0x1004FF43
```
Figure 6-273: `<Xm/VirtKeys.h>`, Part 2 of 2

```
#define osfXK_EndLine 0x1004FF57
#define osfXK_BeginLine 0x1004FF58
#define osfXK_Activate 0x1004FF44
#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
```
```
#define XmUNSPECIFIED_PIXMAP 2
#define XmSTRING_OS_CHARSET XmSTRING_ISO8859_1
#define XmFALLBACK_CHARSET XmSTRING_ISO8859_1
#define XmDEFAULT_FONT _XmSDEFAULT_FONT
#define XmDEFAULT_BACKGROUND _XmSDEFAULT_BACKGROUND
#define XmDEFAULT_DARK_THRESHOLD 20
#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 __XmStringContextRec *XmStringContext;
typedef struct __XmFontListContextRec *XmFontContext;
enum{ XmSTRING_COMPONENT_UNKNOWN, XmSTRING_COMPONENT_CHARSET, XmSTRING_COMPONENT_TEXT, XmSTRING_COMPONENT_DIRECTION, XmSTRING_COMPONENT_SEPARATOR, XmSTRING_COMPONENT_LOCALE_TEXT }
```


```c
#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 _XmPrimitiveRec * XmPrimitiveWidget;

typedef struct _XmGadgetClassRec * XmGadgetClass;
typedef struct _XmGadgetRec * XmGadget;

typedef struct _XmManagerClassRec * XmManagerWidgetClass;
typedef struct _XmManagerRec * XmManagerWidget;

enum { XmCHANGE_ALL, XmCHANGE_NONE, XmCHANGE_WIDTH, XmCHANGE_HEIGHT };
enum { XmPIXLSES, Xm100TH_MILLIMETERS, Xm100TH_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)
```
enum (XmBELLS = 1)
    ;
enum (XmNO_ORIENTATION, XmHORIZONTAL)
    ;
enum (XmWORK_AREA, XmMENU_BAR, XmMENU_PULLDOWN, XmMENU_OPTION)
    ;
enum (XmNO_PACKING, XmPACK_TIGHT, XmPACK_COLUMN, XmPACK_NONE)
    ;
enum (XmALIGNMENT_CONTENTS_TOP = 3, XmALIGNMENT_CONTENTS_BOTTOM)
    ;
enum (XmTEAR_OFF_ENABLED, XmTEAR_OFF_DISABLED)
    ;
enum (XmUNPOST, XmUNPOST_AND_REPLAY)
    ;
enum (XmLAST_POSITION = -1, XmFIRST_POSITION)
    ;
enum (XmALIGNMENT_BEGINNING, XmALIGNMENT_CENTER, XmALIGNMENT_END)
    ;
enum{
  XmALIGNMENT_BASELINE_TOP,
  XmALIGNMENT_BASELINE_BOTTOM = 2,
  XmALIGNMENT_WIDGET_TOP,
  XmALIGNMENT_WIDGET_BOTTOM
};

defined enum{
  XmFRAME_GENERIC_CHILD,
  XmFRAME_TITLE_CHILD
};

defined enum{
  XmN_OF_MANY = 1,
};

defined enum{
  XmATTACH_NONE,
  XmATTACH_OPPOSITE_FORM,
  XmATTACH_OPPOSITE_WIDGET,
  XmATTACH_SELF
};

defined enum{
  XmRESIZE_NONE,
  XmRESIZE_ANY
};
Figure 6-278: <Xm/Xm.h>*, Part 5 of 15

```c
enum{
    XmCR_NONE,
    XmCR_VALUE_CHANGED,
    XmCR_DECREMENT,
    XmCR_PAGE_DECREMENT,
    XmCR_TO_BOTTOM,
    XmCR_ACTIVATE,
    XmCR_DISARM,
    XmCR_UNMAP,
    XmCR_LOSING_FOCUS,
    XmCR_MOVING_INSERT_CURSOR,
    XmCR_SINGLE_SELECT,
    XmCR_EXTENDED_SELECT,
    XmCR_DEFAULT_ACTION,
    XmCR_CLIPBOARD_DATA_DELETE,
    XmCR_OK,
    XmCR_APPLY = 34,
    XmCR_COMMAND_ENTERED,
    XmCR_EXPOSE,
    XmCR_INPUT,
    XmCR_LOSE_PRIMARY,
    XmCR_TEAR_OFF.ACTIVATE,
    XmCR_OBSCURED_TRAVERSAL
} ;

typedef struct
{
    int    reason;
    XEvent *event;
} XmAnyCallbackStruct;

typedef struct
{
    int    reason;
    XEvent *event;
    int    click_count;
} XmArrowButtonCallbackStruct;
```

System Data Interfaces 6-225
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;
    char *data;
    char *callbackstruct;
} XmRowColumnCallbackStruct;
typedef struct
{
    int reason;
    XEvent * event;
    int value;
    int pixel;
} XmScrollBarCallbackStruct;

typedef struct
{
    int reason;
    XEvent * event;
    int set;
} XmToggleButtonCallbackStruct;

typedef struct
{
    int reason;
    XEvent * event;
    XmString item;
    int item_length;
    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;
    int length;
} XmSelectionBoxCallbackStruct;
typedef struct
{
    int reason;
    XEvent *event;
    XmString value;
    int length;
} XmCommandCallbackStruct;

typedef struct
{
    int reason;
    XEvent *event;
    XmString value;
    int length;
    XmString mask;
    int mask_length;
    XmString dir;
    int dir_length;
    XmString pattern;
    int pattern_length;
} XmFileSelectionBoxCallbackStruct;

typedef struct
{
    int reason;
    XEvent *event;
    int value;
} XmScaleCallbackStruct;

eenum{ XmMULTICLICK_DISCARD, XmMULTICLICK_KEEP };

eenum{ XmSHADOW_IN = 7, XmSHADOW_OUT };

Figure 6-281: <Xm/Xm.h>*, Part 8 of 14


<table>
<thead>
<tr>
<th>enum</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XmARROW_UP,</td>
<td></td>
<td>XmARROW_DOWN,</td>
</tr>
<tr>
<td>XmARROW_LEFT,</td>
<td></td>
<td>XmARROW_RIGHT</td>
</tr>
<tr>
<td>XmNO_LINE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmDOUBLE_LINE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmDOUBLE_DASHED_LINE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmSHADOWETCHED_OUT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmSHADOWETCHED_OUT_DASH,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmMAPPED,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmSTATIC,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmVARIABLE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmRESIZE_IF_POSSIBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmAUTOMATIC,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmPIXMAP = 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmWINDOW,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmCURSOR = 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmMAX_ON_TOP,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmMAX_ON_LEFT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmMAX_ON_BOTTOM,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmMAX_ON_RIGHT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmSINGLE_SELECT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmEXTENDED_SELECT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmMULTIPLE_SELECT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmBROWSE_SELECT,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmAPPLICATION_DEFINED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmSTRING,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmSTATIC,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmVARIABLE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmRESIZE_IF_POSSIBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XmAUTOMATIC,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

System Data Interfaces

DRAFT COPY
March 19, 1997
File: abi_386/chap6 (Delta 44.21)
386:adm.book:sum
Page: 320
enum( XmAS_NEEDED = 1
      )
#define SW_TOP 1
#define SW_BOTTOM 0
#define SW_LEFT 2
#define SW_RIGHT 0
#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
      )

ewm( XmMULTI_LINE_EDIT, XmSINGLE_LINE_EDIT
      )

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
typedef enum{
    XmSELECT_POSITION, XmSELECT_WHITESPACE,
    XmSELECT_WORD, XmSELECT_LINE,
    XmSELECT_ALL, XmSELECT_PARAGRAPH
} XmTextScanType;

typedef enum{
    XmHIGHLIGHT_NORMAL, XmHIGHLIGHTSELECTED,
    XmHIGHLIGHTSECONDARYSELECTED
} 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;
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 2

enum { XmDIALOG_NONE, XmDIALOG_APPLY_BUTTON, 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_DIR_LIST, XmDIALOG_RESET, XmDIALOG_SEPARATOR, XmDIALOG_DIR_LABEL, 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
enum { XmDIAGL_OELESS, XmDIAGL_PRIMARY_APPLICATION_MODAL, 
XmDIAGL_FUL APPLICATION_MODAL, XmDIAGL_SYS TEM MODAL
} ;

#define XmDIAGL_APPLICATION_MODAL XmDIAGL_PRIMARY_APPLICATION_MODAL

enum { XmPLACE_TOP, XmPLACE ABOVE SELECTION, 
XmPLACE BELOW SELECTION
} ;

#define XmFILE DIRECTORY (1 << 0)
#define XmFILE REGULAR (1 << 1)
#define XmFILE ANY _TYPE (XmFILE_DIRECTORY | XmFILE_REGULAR)

enum { XmDIAGL_WORK_AREA, XmDIAGL_PROMPT, 
XmDIAGL_SELECTION, XmDIAGL_COMMAND, 
XmDIAGL_FILE_SELECTION
} ;

enum { XmDIAGL_TEMPLATE, XmDIAGL_ERROR, 
XmDIAGL_INFORMATION, XmDIAGL_MESSAGE, 
XmDIAGL_QUESTION, XmDIAGL_WARNING, 
XmDIAGL_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 ;
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;

enum{
    XmPUSHBUTTON = 1,
    XmTOGGLEBUTTON,
    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;
Figure 6-288: <Xm/XmStrDefs.h>, Part 1 of 34

```c
#define XmS ""
#define XmCAccelerator "Accelerator"
#define XmCAcceleratorText "AcceleratorText"
#define XmCJustLast "JustLast"
#define XmCJustMargin "JustMargin"
#define XmCAlignment "Alignment"
#define XmCAAllowOverlap "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 XmCAattachment "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"
```

System Data Interfaces 6-235
#define XmButtons "Buttons"
#define XmCancelButtonLabelString "CancelButtonString"
#define XmChildHorizontalAlignment "ChildHorizontalAlignment"
#define XmChildHorizontalSpacing "ChildHorizontalSpacing"
#define XmChildPlacement "ChildPlacement"
#define XmChildType "ChildType"
#define XmChildVerticalAlignment "ChildVerticalAlignment"
#define XmChildren "Children"
#define XmClientData "ClientData"
#define XmClipWindow "ClipWindow"
#define XmColumns "Columns"
#define XmCommandWindow "CommandWindow"
#define XmCommandWindowLocation "CommandWindowLocation"
#define XmConvertProc "ConvertProc"
#define XmCursorBackground "CursorBackground"
#define XmCursorForeground "CursorForeground"
#define XmCursorPosition "CursorPosition"
#define XmCursorPositionVisible "CursorPositionVisible"
#define XmDarkThreshold "DarkThreshold"
#define XmDecimalPoints "DecimalPoints"
#define XmDefaultButtonShadowThickness "DefaultButtonShadowThickness"
#define XmDefaultButtonType "DefaultButtonType"
#define XmDefaultCopyCursorIcon "DefaultCopyCursorIcon"
#define XmDefaultFontList "DefaultFontList"
#define XmDefaultInvalidCursorIcon "DefaultInvalidCursorIcon"
#define XmDefaultLinkCursorIcon "DefaultLinkCursorIcon"
#define XmDefaultMoveCursorIcon "DefaultMoveCursorIcon"
#define XmDefaultNoneCursorIcon "DefaultNoneCursorIcon"
#define XmDefaultPosition "DefaultPosition"
#define XmDefaultSourceCursorIcon "DefaultSourceCursorIcon"
#define XmDefaultValidCursorIcon "DefaultValidCursorIcon"
#define XmDeleteResponse "DeleteResponse"
#define XmDesktopParent "DesktopParent"
#define XmDialogStyle "DialogStyle"
#define XmDialogTitle "DialogTitle"
Figure 6-290: `<Xm/XmStrDefs.h>`, Part 3 of 34

```
#define XmDialogType "DialogType"
#define XmDirItemCount "DirItemCount"
#define XmDirItems "DirItems"
#define XmDirLabelString "DirLabelString"
#define XmDirMask "DirMask"
#define XmDirSearchProc "DirSearchProc"
#define XmDirSpec "DirSpec"
#define XmDirectory "Directory"
#define XmDirectoryValid "DirectoryValid"
#define XmDisarmCallback "DisarmCallback"
#define XmDoubleClickInterval "DoubleClickInterval"
#define XmDragContextClass "DragContextClass"
#define XmDragDropFinishCallback "DragDropFinishCallback"
#define XmDragIconClass "DragIconClass"
#define XmDragInitiatorProtocolStyle "DragInitiatorProtocolStyle"
#define XmDragMotionCallback "DragMotionCallback"
#define XmDragOperations "DragOperations"
#define XmDragOverMode "DragOverMode"
#define XmDragProc "DragProc"
#define XmDragReceiverProtocolStyle "DragReceiverProtocolStyle"
#define XmDragDropProc "DragDropProc"
#define XmDragRectangles "DragRectangles"
#define XmDropSiteActivity "DropSiteActivity"
#define XmDropSiteEnterCallback "DropSiteEnterCallback"
#define XmDropSiteLeaveCallback "DropSiteLeaveCallback"
#define XmDropSiteManagerClass "DropSiteManagerClass"
#define XmDropSiteOperations "DropSiteOperations"
#define XmDropSiteType "DropSiteType"
#define XmDropStartCallback "DropStartCallback"
#define XmDropTransferClass "DropTransferClass"
#define XmDropTransfers "DropTransfers"
#define XmEditable "Editable"
#define XmEntryBorder "EntryBorder"
#define XmEntryClass "EntryClass"
#define XmExportTargets "ExportTargets"
```
#define XmExposeCallback "ExposeCallback"
#define XmExtensionType "ExtensionType"
#define XmFileListItemCount "FileListItemCount"
#define XmFileListItems "FileListItems"
#define XmFileListLabelString "FileListLabelString"
#define XmFileSearchProc "FileSearchProc"
#define XmFileTypeMask "FileTypeMask"
#define XmFillOnArm "FillOnArm"
#define XmFillOnSelect "FillOnSelect"
#define XmFilterLabelString "FilterLabelString"
#define XmFontList "FontList"
#define XmForegroundThreshold "ForegroundThreshold"
#define XmHelpLabelString "HelpLabelString"
#define XmHighlightColor "HighlightColor"
#define XmHighlightOnEnter "HighlightOnEnter"
#define XmHighlightPixmap "HighlightPixmap"
#define XmHighlightThickness "HighlightThickness"
#define XmHorizontalFontUnit "HorizontalFontUnit"
#define XmHorizontalScrollBar "HorizontalScrollBar"
#define XmHot "Hot"
#define XmICCHandle "ICCHandle"
#define XmImportTargets "ImportTargets"
#define XmIncrement "Increment"
#define XmIncremental "Incremental"
#define XmIndicatorOn "IndicatorOn"
#define XmIndicatorSize "IndicatorSize"
#define XmIndicatorType "IndicatorType"
#define XmInitialDelay "InitialDelay"
#define XmInitialFocus "InitialFocus"
#define XmInputCreate "InputCreate"
#define XmInputMethod "InputMethod"
#define XmInvalidCursorForeground "InvalidCursorForeground"
#define XmIsAligned "IsAligned"
#define XmIsHomogeneous "IsHomogeneous"
#define XmItemCount "ItemCount"
Figure 6-292: <Xm/XmStrDefs.h>, Part 5 of 34

```c
#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"
```
```c
#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"
```
#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"
```c
#define XmCScrolledWindowMarginWidth "ScrolledWindowMarginWidth"
#define XmCScrolledPolicy "ScrollingPolicy"
#define XmCSColor "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 XmCSHows "ShowArrows"
#define XmCSHowsAsDefault "ShowAsDefault"
#define XmCSHowsSeparator "ShowSeparator"
#define XmCSHowsValue "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 XmCSSourceCursorIcon "SourceCursorIcon"
#define XmCSSourceIsExternal "SourceIsExternal"
#define XmCSSourcePixmapIcon "SourcePixmapIcon"
#define XmCSSourceWidget "SourceWidget"
#define XmCSSourceWindow "SourceWindow"
```
#define XmCSSpacing "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 XmCUnselectAllPixmap "UnselectAllPixmap"
#define XmCUpscrollDistance "UpscrollDistance"
#define XmCUntoggle "Untoggle"
#define XmCUserCallback "UserCallback"
#define XmCUserData "UserData"
#define XmCValidCursorForeground "ValidCursorForeground"
#define XmCValueChangedCallback "ValueChangedCallback"
#define XmCValueWcs "ValueWcs"
#define XmCVerifyBell "VerifyBell"
#define XmCVerticalAlignment "VerticalAlignment"
#define XmCVerticalFontUnit "VerticalFontUnit"
#define XmCVerticalScrollBar "VerticalScrollBar"
# 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"
```c
#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"
```
#define XmNcursorForeground "cursorForeground"
#define XmNcursorPosition "cursorPosition"
#define XmNcursorPositionVisible "cursorPositionVisible"
#define XmNdarkThreshold "darkThreshold"
#define XmNdecimalPoints "decimalPoints"
#define XmNdecrementCallback "decrementCallback"
#define XmNdefaultActionCallback "defaultActionCallback"
#define XmNdefaultButtonDown "defaultButtonDown"
#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"
#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 XmNdragStartCallback "dragStartCallback"
#define XmNdragProc "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 XmNentryHorizontalAlignment "entryHorizontalAlignment"
#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

```c
#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 "niccHandle"
define XmNimportTargets "importTargets"
define XmNincrement "increment"
define XmNincrementCallback "incrementCallback"
define XmNincremental "incremental"
define XmNindicatorOn "indicatorOn"
define XmNindicatorSize "indicatorSize"
define XmNindicatorType "indicatorType"
```
# 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

```c
#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"
```
# define XmNmwmFunctions "mwmFunctions"
# define XmNmwmInputMode "mwmInputMode"
# define XmNmwmMenu "mwmMenu"
# define XmNmwmMessages "mwmMessages"
# define XmNavigationType "navigationType"
# define XmNeedsMotion "needsMotion"
# define XmNoMatchCallback "noMatchCallback"
# define XmNoMatchString "noMatchString"
# define XmNoResize "noResize"
# define XmNoneCursorForeground "noneCursorForeground"
# define XmNotifyProc "notifyProc"
# define XmNumColumns "numColumns"
# define XmNumDropRectangles "numDropRectangles"
# define XmNumDropTransfers "numDropTransfers"
# define XmNumExportTargets "numExportTargets"
# define XmNumImportTargets "numImportTargets"
# define XmNumRectangles "numRectangles"
# define XmOffsetX "offsetX"
# define XmOffsetY "offsetY"
# define XmOkCallback "okCallback"
# define XmOkLabelString "okLabelString"
# define XmOperationChangedCallback "operationChangedCallback"
# define XmOperationCursorIcon "operationCursorIcon"
# define XmOptionLabel "optionLabel"
# define XmOptionMnemonic "optionMnemonic"
# define XmOutputCreate "outputCreate"
# define XmPacking "packing"
# define XmPageDecrementCallback "pageDecrementCallback"
# define XmPageIncrement "pageIncrement"
# define XmPageIncrementCallback "pageIncrementCallback"
# define XmPaneMaximum "paneMaximum"
# define XmPaneMinimum "paneMinimum"
# define XmPattern "pattern"
# define XmPendingDelete "pendingDelete"
# define XmPopupEnabled "popupEnabled"
#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 XmNqunifySearchDataProc "qualifySearchDataProc"
#define XmNradioAlwaysOne "radioAlwaysOne"
#define XmNradioBehavior "radioBehavior"
#define XmNrealizeCallback "realizeCallback"
#define XmNrecomputeSize "recomputeSize"
#define XmNrectangle "rectangles"
#define XmNreflectMode "reflectMode"
#define XmNrepeatDelay "repeatDelay"
#define XmNresizeable "resizeable"
#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 XmNsashWidth "sashWidth"
#define XmNscaleHeight "scaleHeight"
#define XmNscaleMultiple "scaleMultiple"
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"
```c
#define XmNsourc "source"
#define XmNsourcCursorIcon "sourceCursorIcon"
#define XmNsourcIsExternal "sourceIsExternal"
#define XmNsourcPixmapIcon "sourcePixmapIcon"
#define XmNsourcWidget "sourceWidget"
#define XmNsourcWindow "sourceWindow"
#define XmNsSpacing "spacing"
#define XmNsSpotLocation "spotLocation"
#define XmNstartTime "startTime"
#define XmNstateCursorIcon "stateCursorIcon"
#define XmNstringDirection "stringDirection"
#define XmNsSubMenuId "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 XmNtopAttachmen "topAttachmen"
#define XmNtopCharacter "topCharacter"
#define XmNtopItemPosition "topItemPosition"
#define XmNtopLevelEnterCallback "topLevelEnterCallback"
#define XmNtopLevelLeaveCallback "topLevelLeaveCallback"
#define XmNtopOffset "topOffset"
#define XmNtopPosition "topPosition"
#define XmNtopShadowColor "topShadowColor"
#define XmNtopShadowPixmap "topShadowPixmap"
```
Figure 6-308: <Xm/XmStrDefs.h>, Part 21 of 34

```c
#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 XmUnmapCallback "unmapCallback"
#define XmUnpostBehavior "unpostBehavior"
#define XmUnselectPixmap "unselectPixmap"
#define XmUpdateSliderSize "updateSliderSize"
#define XmUseAsyncGeometry "useAsyncGeometry"
#define XmUserData "userData"
#define XmValidCursorForeground "validCursorForeground"
#define XmValueChangedCallback "valueChangedCallback"
#define XmValueWcs "valueWcs"
#define XmVerifyBell "verifyBell"
#define XmVerticalFontUnit "verticalFontUnit"
#define XmVerticalScrollBar "verticalScrollBar"
#define XmVerticalSpacing "verticalSpacing"
#define XmVisibleItemCount "visibleItemCount"
#define XmVisibleWhenOff "visibleWhenOff"
#define XmVisualPolicy "visualPolicy"
#define XmWhichButton "whichButton"
#define XmWordWrap "wordWrap"
#define XmworkWindow "workWindow"
#define XmAlignment "Alignment"
#define XmAnimationMask "AnimationMask"
#define XmAnimationPixmap "AnimationPixmap"
#define XmAnimationStyle "AnimationStyle"
#define XmArrowDirection "ArrowDirection"
#define XmAtomList "AtomList"
```
# 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 XmRDialoStyle "DialogStyle"
# define XmRDialoType "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"
Figure 6-310: `<Xm/XmStrDefs.h>`, Part 23 of 34

```c
#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"
```
#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 XmRSizerPolicy "SizePolicy"
#define XmRStringDirection "StringDirection"
#define XmRTearOffModel "TearOffModel"
#define XmRTopShadowPixmap "TopShadowPixmap"
#define XmRTransferStatus "TransferStatus"
#define XmRTraversalType "TraversalType"
#define XmRUnitType "UnitType"
#define XmRUunpostBehavior "UnpostBehavior"
#define XmRVValueWcs "ValueWcs"
#define XmRVVerticalAlignment "VerticalAlignment"
#define XmRVVerticalDimension "VerticalDimension"
#define XmRVVerticalInt "VerticalInt"
#define XmRVVerticalPosition "VerticalPosition"
#define XmRVVirtualBinding "VirtualBinding"
#define XmRVVisibleItemCount "VisibleItemCount"
#define XmRVVisualPolicy "VisualPolicy"
#define XmRWWhichButton "WhichButton"
#define XmRXmBackgroundPixmap "XmBackgroundPixmap"
#define XmRXmString "XmString"
#define XmRXmStringCharSet "XmStringCharSet"
#define XmRXmStringTable "XmStringTable"
#define XmVsosfActivate "osfActivate"
#define XmVsosfAddMode "osfAddMode"
#define XmVsosfBackSpace "osfBackSpace"
#define XmVsosfBeginLine "osfBeginLine"
#define XmVsosfCancel "osfCancel"
#define XmVsosfClear "osfClear"
System Data Interfaces

6-259

DRAFT COPY
March 19, 1997
File: abi_386/chap6 (Delta 44.21)
386:adm.book:sum

---

Figure 6-312: `<Xm/XmStrDefs.h>`, Part 25 of 34

```c
#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 XmSFontList_Default_Tag XmSFontList_Default_Tag_STRING
#define XmSxmFontList_Default_Tag_STRING
#define XmVaCASCADEBUTTON "cascadeButton"
#define XmVaCHECKBUTTON "checkButton"
#define XmVaDOUBLE_SEPARATOR "doubleSeparator"
#define XmVaPUSHBUTTON "pushButton"
#define XmVaRADIOBUTTON "radioButton"
```
# define Xm Va SEPARATOR "separator"
# define Xm Va SINGLE_ SEPARATOR "singleSeparator"
# define Xm Va TOGGLEBUTTON "checkButton"
# define Xm Va TITLE XtNtitle

# define Xt CKeyboardFocusPolicy XmCKeyboardFocusPolicy
# define Xt CShellUnitType XmCShellUnitType
# define Xt NkeyboardFocusPolicy XmNkeyboardFocusPolicy
# define Xt NshellUnitType XmNshellUnitType
# define Xt RkeyboardFocusPolicy XmRkeyboardFocusPolicy

# define Xm RPrimBottomShadowPixmap XmRBottomShadowPixmap
# define Xm RPrimHighlightPixmap XmRHighlightPixmap
# define Xm RPrimTopShadowPixmap XmRTopShadowPixmap

# define Xm CAccelerators XtCAccelerators
# define Xm CallowShellResize XtCallowShellResize
# define Xm CArgc XtCargc
# define Xm CArgv XtCargv
# define Xm CBackground XtCbackground
# define Xm CBaseHeight XtCbaseHeight
# define Xm CBaseHeight XtCbaseHeight
# define Xm CBaseWidth XtCbaseWidth
# define Xm CBaseWidth XtCbaseWidth
# define Xm CBitmap XtCBitmap
# define Xm CBoolean XtCBoolean
# define Xm CBorderColor XtCBorderColor
# define Xm CBorderWidth XtCBorderWidth
# define Xm CCallback XtCCallback
# define Xm CColor XtCColor
# define Xm CColorMap XtCColorMap
# define Xm CCreatePopupChildProc XtCCreatePopupChildProc
# define Xm CCursor XtCCursor
# define Xm CDepth XtCDepth
# define Xm CDimension XtRDimension
```
#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 XmCFunctor        XtCFunctor
#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
#define XmCInitialResourcesPersistent XtCInitialResourcesPersistent
#define XmCInitialState   XtCInitialState
#define XmCInput          XtCInput
#define XmCInsertPosition XtCInsertPosition
#define XmCInterval       XtCInterval
#define XmCJustify        XtCJustify
#define XmCLabel          XtCLabel
#define XmCLength         XtCLength
#define XmCMappedWhenManaged XtCMappedWhenManaged
#define XmCMargin         XtCMargin
#define XmCMaxAspectX     XtCMaxAspectX
#define XmCMaxAspectY     XtCMaxAspectY
#define XmCMaxHeight      XtCMaxHeight
```
<table>
<thead>
<tr>
<th>Definition</th>
<th>C Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>XmCMaxWidth</td>
<td>XtCMaxWidth</td>
</tr>
<tr>
<td>XmCMenuEntry</td>
<td>XtCMenuEntry</td>
</tr>
<tr>
<td>XmCMinAspectX</td>
<td>XtCMinAspectX</td>
</tr>
<tr>
<td>XmCMinAspectY</td>
<td>XtCMinAspectY</td>
</tr>
<tr>
<td>XmCMinHeight</td>
<td>XtCMinHeight</td>
</tr>
<tr>
<td>XmCMinWidth</td>
<td>XtCMinWidth</td>
</tr>
<tr>
<td>XmCNotify</td>
<td>XtCNotify</td>
</tr>
<tr>
<td>XmCOrientation</td>
<td>XtCOrientation</td>
</tr>
<tr>
<td>XmCOVERRIDE_REDIRECT</td>
<td>XtCOverrideRedirect</td>
</tr>
<tr>
<td>XmCParameter</td>
<td>XtCParameter</td>
</tr>
<tr>
<td>XmCPixmap</td>
<td>XtCPixmap</td>
</tr>
<tr>
<td>XmCPosition</td>
<td>XtCPosition</td>
</tr>
<tr>
<td>XmCReadOnly</td>
<td>XtCReadOnly</td>
</tr>
<tr>
<td>XmCResize</td>
<td>XtCResize</td>
</tr>
<tr>
<td>XmCReverseVideo</td>
<td>XtCReverseVideo</td>
</tr>
<tr>
<td>XmCSaveUnder</td>
<td>XtCSaveUnder</td>
</tr>
<tr>
<td>XmCScreen</td>
<td>XtCScreen</td>
</tr>
<tr>
<td>XmCScrollDCursor</td>
<td>XtCScrollDCursor</td>
</tr>
<tr>
<td>XmCScrollHCursor</td>
<td>XtCScrollHCursor</td>
</tr>
<tr>
<td>XmCScrollLCursor</td>
<td>XtCScrollLCursor</td>
</tr>
<tr>
<td>XmCScrollProc</td>
<td>XtCScrollProc</td>
</tr>
<tr>
<td>XmCScrollRCursor</td>
<td>XtCScrollRCursor</td>
</tr>
<tr>
<td>XmCScrollUCursor</td>
<td>XtCScrollUCursor</td>
</tr>
<tr>
<td>XmCScrollVCursor</td>
<td>XtCScrollVCursor</td>
</tr>
<tr>
<td>XmCSelection</td>
<td>XtCSelection</td>
</tr>
<tr>
<td>XmCSelectionArray</td>
<td>XtCSelectionArray</td>
</tr>
<tr>
<td>XmCSensitive</td>
<td>XtCSensitive</td>
</tr>
<tr>
<td>XmCspace</td>
<td>XtCspace</td>
</tr>
<tr>
<td>XmCstring</td>
<td>XtCstring</td>
</tr>
<tr>
<td>XmCTextOptions</td>
<td>XtCTextOptions</td>
</tr>
<tr>
<td>XmCTextPosition</td>
<td>XtCTextPosition</td>
</tr>
<tr>
<td>XmCTextSink</td>
<td>XtCTextSink</td>
</tr>
<tr>
<td>XmCTextSource</td>
<td>XtCTextSource</td>
</tr>
<tr>
<td>XmCThickness</td>
<td>XtCThickness</td>
</tr>
<tr>
<td>XmCThumb</td>
<td>XtCThumb</td>
</tr>
</tbody>
</table>
Figure 6-316: `<Xm/XmStrDefs.h>`, Part 29 of 34

<table>
<thead>
<tr>
<th>#define XmCTitle</th>
<th>XtCTitle</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define XmCTitleEncoding</td>
<td>XtCTitleEncoding</td>
</tr>
<tr>
<td>#define XmCTransient</td>
<td>XtCTransient</td>
</tr>
<tr>
<td>#define XmCTransientFor</td>
<td>XtCTransientFor</td>
</tr>
<tr>
<td>#define XmCTranslations</td>
<td>XtCTranslations</td>
</tr>
<tr>
<td>#define XmCVSpace</td>
<td>XtCVSpace</td>
</tr>
<tr>
<td>#define XmCValue</td>
<td>XtCValue</td>
</tr>
<tr>
<td>#define XmCVSVisual</td>
<td>XtCVVisual</td>
</tr>
<tr>
<td>#define XmCWaitForWm</td>
<td>XtCWaitForWm</td>
</tr>
<tr>
<td>#define XmCWidget</td>
<td>XtCWidget</td>
</tr>
<tr>
<td>#define XmCWidth</td>
<td>XtCWidth</td>
</tr>
<tr>
<td>#define XmCWidthInc</td>
<td>XtCWidthInc</td>
</tr>
<tr>
<td>#define XmCWinGravity</td>
<td>XtCWinGravity</td>
</tr>
<tr>
<td>#define XmCWindow</td>
<td>XtCWindow</td>
</tr>
<tr>
<td>#define XmCWindowGroup</td>
<td>XtCWindowGroup</td>
</tr>
<tr>
<td>#define XmCWmTimeout</td>
<td>XtCWmTimeout</td>
</tr>
<tr>
<td>#define XmCX</td>
<td>XtCX</td>
</tr>
<tr>
<td>#define XmCY</td>
<td>XtCY</td>
</tr>
<tr>
<td>#define XmNaccelerators</td>
<td>XtNaccelerators</td>
</tr>
<tr>
<td>#define XmNallowShellResize</td>
<td>XtNallowShellResize</td>
</tr>
<tr>
<td>#define XmNancestorsensitive</td>
<td>XtNancestorsensitive</td>
</tr>
<tr>
<td>#define XmNargc</td>
<td>XtNargc</td>
</tr>
<tr>
<td>#define XmNargv</td>
<td>XtNargv</td>
</tr>
<tr>
<td>#define XmNbackground</td>
<td>XtNbackground</td>
</tr>
<tr>
<td>#define XmNbackgroundPixmap</td>
<td>XtNbackgroundPixmap</td>
</tr>
<tr>
<td>#define XmNbaseHeight</td>
<td>XtNbaseHeight</td>
</tr>
<tr>
<td>#define XmNbaseHeight</td>
<td>XtNbaseHeight</td>
</tr>
<tr>
<td>#define XmNbaseWidth</td>
<td>XtNbaseWidth</td>
</tr>
<tr>
<td>#define XmNbaseWidth</td>
<td>XtNbaseWidth</td>
</tr>
<tr>
<td>#define XmNbitmap</td>
<td>XtNbitmap</td>
</tr>
<tr>
<td>#define XmNborder</td>
<td>XtNborder</td>
</tr>
<tr>
<td>#define XmNborderColor</td>
<td>XtNborderColor</td>
</tr>
<tr>
<td>#define XmNborderPixmap</td>
<td>XtNborderPixmap</td>
</tr>
<tr>
<td>#define XmNborderWidth</td>
<td>XtNborderWidth</td>
</tr>
<tr>
<td>#define XmNcallback</td>
<td>XtNcallback</td>
</tr>
</tbody>
</table>
Figure 6-317: \(<Xm/XmStrDefs.h>\)*, Part 30 of 34

```
#define XmNchildren          XtNchildren
#define XmNcolorormap        XtNcolorormap
#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
#define XmNinternalHeight    XtNinternalHeight
#define XmNinternalWidth     XtNinternalWidth
#define XmNjumpProc          XtNjumpProc
```
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 Xm NORientation XtNorientation
#define XmNoverrideRedirect XtNoverrideRedirect
#define XmNparameter XtNparameter
#define XmNpixmap XtNpixmap
#define XmNpopupCallback XtNpopupCallback
#define XmNresize XtNresize
#define XmNreverseVideo XtNreverseVideo
#define XmNs��vUnder XtNs��vUnder
#define XmNs��n XtNs��n
#define XmNscrollDCursor XtNscrollDCursor
#define XmNscrollHCursor XtNscrollHCursor
#define XmNscrollLCursor XtNscrollLCursor
#define XmNscrollProc XtNscrollProc
#define XmNscrollRCursor XtNscrollRCursor
#define XmNscrollUCursor XtNscrollUCursor
#define XmNscrollVCursor XtNscrollVCursor
#define XmNselection XtNselection
#define XmNselectionArray XtNselectionArray
```

System Data Interfaces 6-265
<table>
<thead>
<tr>
<th>#define XmNsensitive</th>
<th>XtNsensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define XmNshown</td>
<td>XtNshown</td>
</tr>
<tr>
<td>#define XmNsensitive</td>
<td>XtNsensitive</td>
</tr>
<tr>
<td>#define XmNstring</td>
<td>XtNstring</td>
</tr>
<tr>
<td>#define XmNTextOptions</td>
<td>XtNTextOptions</td>
</tr>
<tr>
<td>#define XmNTextSink</td>
<td>XtNTextSink</td>
</tr>
<tr>
<td>#define XmNTextSource</td>
<td>XtNTextSource</td>
</tr>
<tr>
<td>#define XmNthickness</td>
<td>XtNthickness</td>
</tr>
<tr>
<td>#define XmNthumb</td>
<td>XtNthumb</td>
</tr>
<tr>
<td>#define XmNthumbProc</td>
<td>XtNthumbProc</td>
</tr>
<tr>
<td>#define XmNtitle</td>
<td>XtNtitle</td>
</tr>
<tr>
<td>#define XmNtitleEncoding</td>
<td>XtNtitleEncoding</td>
</tr>
<tr>
<td>#define XmNtop</td>
<td>XtNtop</td>
</tr>
<tr>
<td>#define XmNtransient</td>
<td>XtNtransient</td>
</tr>
<tr>
<td>#define XmNtransientFor</td>
<td>XtNtransientFor</td>
</tr>
<tr>
<td>#define XmNtransient</td>
<td>XtNtransient</td>
</tr>
<tr>
<td>#define XmNtransientFor</td>
<td>XtNtransientFor</td>
</tr>
<tr>
<td>#define XmNtranslations</td>
<td>XtNtranslations</td>
</tr>
<tr>
<td>#define XmNupdate</td>
<td>XtNupdate</td>
</tr>
<tr>
<td>#define XmNuseBottom</td>
<td>XtNuseBottom</td>
</tr>
<tr>
<td>#define XmNuseRight</td>
<td>XtNuseRight</td>
</tr>
<tr>
<td>#define XmNvalue</td>
<td>XtNvalue</td>
</tr>
<tr>
<td>#define XmNvisual</td>
<td>XtNvisual</td>
</tr>
<tr>
<td>#define XmNwaitForWm</td>
<td>XtNwaitForWm</td>
</tr>
<tr>
<td>#define XmNwidth</td>
<td>XtNwidth</td>
</tr>
<tr>
<td>#define XmNwidthInc</td>
<td>XtNwidthInc</td>
</tr>
<tr>
<td>#define XmNwinGravity</td>
<td>XtNwinGravity</td>
</tr>
<tr>
<td>#define XmNwindow</td>
<td>XtNwindow</td>
</tr>
<tr>
<td>#define XmNwindowGroup</td>
<td>XtNwindowGroup</td>
</tr>
<tr>
<td>#define XmNwmTimeout</td>
<td>XtNwmTimeout</td>
</tr>
<tr>
<td>#define XmNx</td>
<td>XtNx</td>
</tr>
<tr>
<td>#define XmNy</td>
<td>XtNy</td>
</tr>
<tr>
<td>#define XmRAcceleratorTable</td>
<td>XtRAcceleratorTable</td>
</tr>
<tr>
<td>#define XmRAtom</td>
<td>XtRAtom</td>
</tr>
<tr>
<td>#define XmRBitmap</td>
<td>XtRBitmap</td>
</tr>
<tr>
<td>#define XmRBool</td>
<td>XtRBool</td>
</tr>
</tbody>
</table>
**Figure 6-320:** `<Xm/XmStrDefs.h>`, Part 33 of 34

```c
#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 XmRFONTSSTRUCT       XtRFONTSSTRUCT    
#define XmRFUNCTION          XtRFUNCTION      
#define XmRGEOMETRY          XtRGEOMETRY      
#define XmRIMMEDIATE         XtRIMMEDIATE     
#define XmRINITIALSTATE      XtRINITIALSTATE   
#define XmRINT               XtRINT            
#define XmRJUSTIFY           XtRJUSTIFY       
#define XmRLONGBOOLEAN       XtRLONGBOOLEAN   
#define XmROREIENTATION      XtROREIENTATION  
#define XmROBJECT            XtROBJECT        
#define XmRPIXEL             XtRPIXEL          
#define XmRPIXMAP            XtRPIXMAP         
#define XmRPOINTER           XtRPOINTER       
#define XmRPOSITION          XtRPOSITION      
#define XmRSCREEN            XtRSCREEN        
#define XmRSHORT             XtRSHORT          
#define XmRSTRING            XtRSTRING        
#define XmRSTRINGARRAY       XtRSTRINGARRAY   
#define XmRSTRINGTABLE       XtRSTRINGTABLE   
#define XmRTEXTPOSITION      XtRTEXTPOSITION  
```

System Data Interfaces

6-267
Figure 6-321: `<Xm/XmStrDefs.h>`, Part 34 of 34

<table>
<thead>
<tr>
<th>#define XmRTranslationTable</th>
<th>XtRTranslationTable</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define XmRUNsignedChar</td>
<td>XtRUNsignedChar</td>
</tr>
<tr>
<td>#define XmRVisual</td>
<td>XtRVisual</td>
</tr>
<tr>
<td>#define XmRWidget</td>
<td>XtRWidget</td>
</tr>
<tr>
<td>#define XmRWidgetClass</td>
<td>XtRWidgetClass</td>
</tr>
<tr>
<td>#define XmRWidgetList</td>
<td>XtRWidgetList</td>
</tr>
<tr>
<td>#define XmRWindow</td>
<td>XtRWindow</td>
</tr>
</tbody>
</table>
TCP/IP Data Definitions

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.
```c
#define IPPROTO_IP 0
#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

#define IN_SET_LOOPBACK_ADDR(a) {
    (a)->sin_addr.s_addr = htonl(INADDR_LOOPBACK);
    (a)->sin_family = AF_INET;
}

struct sockaddr_in {
    short sin_family;
    u_short sin_port;
    struct in_addr sin_addr;
    char sin_zero[8];
};

#define IP_OPTIONS 1
```
Figure 6-323: <netinet/ip.h>

```c
#define IPOPT_EOL 0
#define IPOPT_NOP 1
#define IPOPT_LSRR 131
#define IPOPT_SSSR 137
```

Figure 6-324: <netinet/tcp.h>

```c
#define TCP_NODELAY 0x01
```
# DEVELOPMENT ENVIRONMENT

## Development Commands

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATH Access to Development Tools</td>
<td>7-1</td>
</tr>
</tbody>
</table>

## Software Packaging Tools

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Headers</td>
<td>7-2</td>
</tr>
<tr>
<td>Static Archives</td>
<td>7-2</td>
</tr>
</tbody>
</table>
Development Commands

THE FACILITIES AND INTERFACES DESCRIBED IN THIS SECTION ARE OPTIONAL COMPONENTS OF THE System V Application Binary Interface.

NOTE

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;

\[
\begin{align*}
as & \quad cc & \quad ld \\
m4 & \quad lex & \quad yacc
\end{align*}
\]

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.
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 environments. The archive libm.a is located in /usr/lib on conforming i386 development environments.
8 EXECUTION ENVIRONMENT

Application Environment 8-1
The /dev Subtree 8-1
Application Environment

NOTE
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 ttynam() 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.
/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:

```plaintext
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:

```plaintext
c#t#l#
c#t#l#
ccdrom#
```

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.
Index

A
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
absolute code 3:34, 5:3
see also position-independent code 3:34
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
scalar types 3:2
stack 3:11
stack frame 3:10, 44
structure and union 3:3
allocation, dynamic stack space
3:43–44
ANSI, C (see C language, ANSI)
ANSI C 3:28
Application Environment 8:1
implementation 3:1
processor 3:1
archive file 7:2
argc 3:26
arguments
bad assumptions 3:44
eexec(BA_OS) 3:26
floating-point 3:17
function 3:9
integer 3:17
main 3:26
order 3:11
pointer 3:17
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

B
base address 3:31, 4:4, 5:3
behavior, undefined (see undefined behavior)
bit-field 3:6
alignment 3:6
allocation 3:6
unnamed 3:6
bounds check fault 3:25
branch instructions 3: 41
breakpoint trap 3: 25
BulletinB.h 6: 193

C
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
calling sequence 3: 9
    function epilogue 3: 13, 15–16, 37
    function prologue 3: 13, 15, 37
    function prologue and epilogue 3: 36
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 parameters)
Constraint.h 6: 131
coprocessor error fault 3: 25
coprocessor overrun abort 3: 25
Core.h 6: 131
cctype.h 6: 7
cursorfont.h 6: 134
CutPaste.h 6: 195

D
data
    process 3: 20
    uninitialized 5: 2
data representation 3: 1
Development Environment 7: 1–2
DialogS.h 6: 195
dirent.h 6: 8
Display.h 6: 196
divide error fault 3: 25
d1fcn.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
DropSMgr.h 6: 204
DropTrans.h 6: 205
dynamic frame size 3: 43
dynamic linking 3: 14, 17, 20, 5: 5
environment 5: 10
    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

E
EFLAGS register, initial value 3: 26
EIP-relative 3: 35, 41
ELF 4: 1
e1f.h 6: 14
emulation, instructions 3: 1
environment 5: 10
    exec(BA_OS) 3: 26
envp 3: 26
erro.h 6: 17
exceptions 3: 24
interface 3: 24
signals 3: 24
eexec(BA_OS) 3: 28, 35
interpreter 3: 31
paging 5: 1
process initialization 3: 26
executable file, segments 5: 3
execution mode (see processor execution mode)
_exit(BA_OS) 3: 24
extended instruction pointer, relative addressing (see EIP-relative)
ex tended-precision 3: 13

F
fcnt1.h 6: 19
file, object (see object file)
file offset 5: 1
FileSB.h 6: 205
float 3: 2
float.h 6: 21
floating-point
arguments 3: 17
return value 3: 12
floating-point control register, initial value 3: 26
floating-point status register, initial value 3: 26
fmtmsg.h 6: 22
fnmatch.h 6: 22
formats
array 3: 3
structure 3: 3
union 3: 3
Form.h 6: 205
FORTRAN
COMMON 3: 3
EQUIVALENCE 3: 3
_fpstart 3: 28
frame pointer 3: 11, 37
initial value 3: 29
frame size, dynamic 3: 44
Frame.h 6: 206
ftw.h 6: 23
function, void 3: 12
function arguments (see arguments)
function call, code 3: 39
function linkage (see calling sequence)
function prologue and epilogue (see calling sequence)

general protection fault/abort 3: 25
global offset table 3: 14, 17, 35, 4: 2, 4–6, 5: 5
ebx 3: 36
_GLOBAL_OFFSET_TABLE_ 3: 36
relocation 3: 35
_GLOBAL_OFFSET_TABLE_ (see global offset table)
glob.h 6: 24
grp.h 6: 25

H
halfword 3: 1
heap, dynamic stack 3: 44

I
iconv.h 6: 25
IEEE 754 3: 28
in.h 6: 270
initialization, process 3: 26
instructions, emulation 3: 1
int 3: 2
int instruction 3: 25
integer arguments 3: 17

Index

IN-3

DRAFT COPY
March 19, 1997
File: abi_386/index (Delta 44.5)
386:adm.book:sum

Page: 372
Intel386  3: 1, 9, 24, 44, 5: 1
Intel387  3: 1
interrupt  3: 12
Intrinsic.h  6: 140
invalid opcode fault  3: 25
invalid TSS fault  3: 25
ip.h  6: 271

J
jmp instruction  5: 8

L
LabelG.h  6: 206
Label.h  6: 206
langinfo.h  6: 28
lazy binding  5: 10
lcall instruction  3: 24
ld  7: 1
LD_BIND_NOW  5: 10
ld(SD_CMD) (see link editor)
lex  7: 1
libc  6: 2
libcurses  7: 2
libm  7: 2
limits.h  6: 30
link editor  4: 6, 5: 5
linkage, function (see calling sequence)
List.h  6: 207
local variables  3: 43
locale.h  6: 31
long  3: 2
long double  3: 2
longjmp(BA_LIB) (see setjmp(BA_LIB))
lwpsynch.h  6: 32

M
m4  7: 1
Machine Status Word register, initial value  3: 26
machlock.h  6: 32
main arguments  3: 26
declaration  3: 26
MainW.h  6: 207
malloc(BA_OS)  3: 22-23
math.h  6: 32
memory allocation, stack  3: 43-44
memory management  3: 20
MenuShell.h  6: 207
MessageB.h  6: 208
mmap(KE_OS)  3: 23
MrmPublic.h  6: 212
MwmUtil.h  6: 214

N
netconfig.h  6: 38
netdir.h  6: 40
nl_types.h  6: 40
no coprocessor fault  3: 25
nonmaskable interrupt  3: 25
null pointer  3: 3, 21, 26

O
object file  4: 1
ELF header  4: 1
executable  3: 35
executable file  3: 35
relocation  4: 4
section  4: 2
see also archive file  4: 1
see also dynamic linking  5: 5
see also executable file  4: 1
see also relocatable file  4: 1
see also shared object file  4: 1
segment  5: 1
shared object file  3: 35
special sections  4: 2
Object.h  6: 140
offset table, global (see global offset table)
optimization  3: 14, 17
overflow trap  3: 25

P
padding
  arguments  3: 11, 18
  structure and union  3: 3
page fault  3: 25
page size  3: 20, 31, 5: 1
paging  3: 20, 5: 1
  performance  5: 1
PanedW.h  6: 214
parameters
  function (see arguments)
  system configuration (see tunable parameters)
PATH variable  7: 1
performance, paging  5: 1
physical addressing  3: 20
pkgmk  7: 2
pkgproto  7: 2
pkgtrans  7: 2
pointer  3: 3
  function argument  3: 17
null  3: 3, 26
poll.h  6: 42
portability
  C program  3: 44
  instructions  3: 1
position-independent code  3: 14, 17,
  34, 36, 5: 3
see also absolute code  3: 34
see also global offset table  3: 35

see also procedure linkage table
3: 35
procct1.h  6: 43
procedure linkage table  3: 35, 4: 2, 5-6,
  5: 5, 7
procedures (see functions)
process
dead  3: 45
entry point  3: 26
initialization  3: 26
segment  3: 20
size  3: 20
stack  3: 28
virtual addressing  3: 20
processor architecture  3: 1
processor execution mode  3: 24, 26
processor-specific information  3: 1, 9,
  20, 34, 5: 1, 5-7, 6: 2-3, 7: 1
program loading  3: 31, 5: 1
PushBG.h  6: 215
PushB.h  6: 215
pushl instruction  5: 8
pwd.h  6: 45

R
RectObj.h  6: 140
re-entrancy  3: 14
regex.h  6: 47
registers
  calling sequence  3: 11
cs, ds, es, and ss  3: 29
description  3: 9, 11
eax  3: 11
ebp  3: 11
ebx  3: 11
ecx  3: 12
edi  3: 11
edx  3: 12
EFLAGS  3: 12
esi  3: 11
esp 3:11
floating-point 3:9, 12
floating-point control word 3:12
floating-point return value 3:13
global 3:9
initial values 3:29
scratch 3:12
variable 3:11
registers cr0 (see Machine Status Word register)
registers eax, integer return value 3:12
relocation
global offset table 3:35
see object file 4:4
RepType.h 6:215
resources, shared 3:20
ret instruction 3:13, 15
return address 3:13
return value
floating-point 3:12
integer 3:12
pointer 3:12
structure and union 3:14
RowColumn.h 6:216
rpc.h 6:63
deptocntl.h 6:63
process 3:20–21, 5:1, 6
segment not present fault 3:25
segment permissions 3:23
segment registers 3:29
SelectioB.h 6:217
SeparatoG.h 6:217
Separator.h 6:217
setjmp(BA_LIB) 3:45
setjmp.h 6:66
setrlimit(BA_OS) 3:23
shared object file 3:35
segments 3:21, 5:3
Shell.h 6:141
short 3:2
sign extension
arguments 3:17
bit-field 3:6
signal(BA_OS) 3:12, 24
signal.h 6:69
signals 3:12, 45
signed 3:2, 6
single step trap/fault 3:25
sizeof 3:2
structure 3:3
Ssape.h 6:141
stack
address 3:29
dynamic allocation 3:44
growth 3:10
initial process 3:28
process 3:20–21
system management 3:22
stack exception fault 3:25
stack frame 3:9–10, 13–15, 17, 37, 43
alignment 3:10, 29, 44
organization 3:10, 43
size 3:11, 43
stack pointer 3:11, 29, 37
initial value 3:29
<stdio.h> 3:44
stdarg.h 6:76
stddef.h 6:76

S
scalar types 3:2
Scale.h 6:216
Screen.h 6:216
ScrollBar.h 6:216
ScrolledList.h 6:217
search.h 6:64
secondary storage 3:20
section, object file 5:1
segment
dynamic 3:21
permissions 5:2

IN-6 Index

DRAFT COPY
March 19, 1997
File: abi_386/index (Delta 44.5)
386:adm.book:sum

Page: 375
stdio.h  6:79
stdlib.h  6:79
stropts.h  6:85
structure  3:3
  function argument  3:18
  padding  3:3
  return value  3:14
supervisor mode (see processor execution mode)
SVID  7:2
switch statements  3:41
<synch>  6:88
sysconf(BA_OS)  3:20, 31
sys/ipc.h  6:26
sys/mman.h  6:33
sys/mod.h  6:34
sys/mount.h  6:35
sys/param.h  6:41
sys/procset.h  6:44
sys/resource.h  6:48
sys/sem.h  6:65
sys/shm.h  6:66
sys/siginfo.h  6:72
sys/stat.h  6:74
sys/statvfs.h  6:75
sys/syscalls.h  6:88
System Headers  7:2
  system load  3:20
System V ABI  6:4
sys/time.h  6:103
sys/times.h  6:104
sys/types.h  6:114
sys/uio.h  6:117
sys/utsname.h  6:120
T
tcip/tcp.h  6:271
termination, process  3:45
termios.h  6:98
text
  process  3:20
  sharing  3:35
TextF.h  6:218
Text.h  6:218
<thread>  6:100
ticlts.h  6:100
ticots.h  6:101
ticotords.h  6:101
time.h  6:102
tiuser.h  6:112
ToggleBG.h  6:218
ToggleB.h  6:218
tspricntl.h  6:113
tunable parameters, stack size  3:23
type mismatch  3:15
U
ucontext.h  6:116
ulimit.h  6:117
unaligned address (see address, unaligned)
undefined behavior  3:1, 12, 15, 28–29, 5:2
  see also ABI conformance  3:1
  see also unspecified property  3:1
uninitialized data  5:2
union  3:3
  function argument  3:18
  return value  3:14
unistd.h  6:119
unsigned  3:2, 6
unspecified property  3:1, 12, 14, 24, 26–27, 29, 5:1–2
  see also ABI conformance  3:1
  see also undefined behavior  3:1
user mode (see processor execution mode)
utime.h  6:119
| 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 |
|   | X11lib.h | 6:184 |
|   | Xm.h | 6:234 |
|   | XmStrDefs.h | 6:266 |
|   | Xresource.h | 6:186 |
|   | Xutil.h | 6:191 |

| Y | yacc | 7:1 |

| Z | zero | |
|   | null pointer | 3:3 |
|   | uninitialized data | 5:2 |