SYSTEM V
APPLICATION
BINARY INTERFACE

Intel386™ Architecture
Processor Supplement

Fourth Edition

ACKNOWLEDGEMENTS

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

TRADEMARKS

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

Contents

Table of Contents

1   INTRODUCTION

2   SOFTWARE INSTALLATION

3   LOW-LEVEL SYSTEM INFORMATION

Figures and Tables

Table of Contents   **vii**

Table of Contents   **ix**

Table of Contents   **xi**

1 INTRODUCTION

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   M
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   M
the Intel386 architecture, this *Intel386 ABI* also applies to any system built with the   M
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)

- *80386 Programmer's Reference Manual* (Intel Literature order number 230985)

- *80387 Programmer's Reference Manual* (Intel Literature order number 231917)

- *UnixWare® 2.0 Command Reference (a-l)*

- *UnixWare® 2.0 Command Reference (m-z)*

- *UnixWare® 2.0 Operating System API Reference: System Calls*

- *UnixWare® 2.0 Operating System API Reference: Library Functions*

- *UnixWare® 2.0 System Administration: Volumes I and II*

- *System V Interface Definition, Edition 4*

NOTE	Diffmarkings have been retained in the text of this book to indicate in which revisions of System V certain modifications were made to the *ABI*. A "G" character in the right hand margin indicates a change in the *ABI* made in UNIX System V Release 4.2. A "M" character in the right hand margin indicates a change in the *ABI* made in UnixWare® 2.0.

M

DRAFT COPY
March 19, 1997
File: abi_386/chap1 (Delta 44.5)
386:adm.book:sum

Page: 17

2 SOFTWARE INSTALLATION

Software Distribution Formats

Physical Distribution Media

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

- 1.44MB 3 1/2" floppy disk: quad-density, double-sided, 80 tracks/side, 18 sectors/track, 512 bytes/sector.

- 1.2MB 5 1/4" floppy disk: quad-density, double-sided, 80 tracks/side, 15 sectors/track, 512 bytes/sector.

- 360KB 5 1/4" floppy disk: double-density, double-sided, 40 tracks/side, 9 sectors/track, 512 bytes/sector.

- 60 MB quarter-inch cartridge tape in QIC-24 format.   G

- CD-ROM optical disks.   G

- 150 MB quarter-inch tape.

▽ CAUTION The use of 360KB 5 1/4" floppy disk, and 60 MB quarter inch cartridge tape   G
           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:

DRAFT COPY
March 19, 1997
File: abi_386/chap2 (Delta 44.3)
386:adm.book:sum

Page: 19

```
#define NICFREE         50
#define NICINOD         100

struct filsys {
        u_short  s_isize;
        daddr_t  s_fsize;
        short            s_nfree;
        daddr_t  s_free[NICFREE];
        short            s_ninode;
        ushort_t s_inode[NICINOD];
        char             s_flock;
        char             s_ilock;
        char             s_fmod;
        char             s_ronly;
        time_t           s_time;
        short            s_dinfo[4];
        daddr_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   0xbadbc14b
```

s_type indicates the file system type. Currently, three types of file systems are supported: the original 512-byte logical block, the 1024-byte logical block, and the 2048-byte logical block. *s_magic* is used to distinguish the original 512-byte oriented file systems from the newer file systems. If this field is not equal to the magic number, fsMAGIC, the type is assumed to be 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_sblkno;
        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_ncg;
        long            fs_bsize;
        long            fs_fsize;
        long            fs_frag;
        long            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_maxbpg;
        long            fs_fragshift;
        long            fs_fsbtodb;
        long            fs_sbsize;
        long            fs_csmask;
        long            fs_csshift;
        long            fs_nindir;
        long            fs_inopb;
        long            fs_nspf;
        long            fs_optim;
        long            fs_state;
        long            fs_sparecon[2];
        long            fs_id[2];
        daddr_t         fs_csaddr;
        long            fs_cssize;
        long            fs_cgsize;
        long            fs_ntrak;
        long            fs_nsect;
        long            fs_spc;
        long            fs_ncyl;
```

DRAFT COPY
March 19, 1997
File: abi_386/chap2 (Delta 44.3)
386:adm.book:sum

Page: 22

```
        long        fs_cpg;
        long        fs_ipg;
        long        fs_fpg;
        struct csum  fs_cstotal;
        char        fs_fmod;
        char        fs_clean;
        char        fs_ronly;
        char        fs_flags;
        char        fs_fsmnt[MAXMNTLEN];
        long        fs_cgrotor;
        struct csum  *fs_csp[MAXCSBUFS];
        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_cgx;
        short       cg_ncyl;
        short       cg_niblk;
        long        cg_ndblk;
        struct csum  cg_cs;
        long        cg_rotor;
        long        cg_frotor;
        long        cg_irotor;
        long        cg_frsum[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 SBSIZE       8192
#define BBLOCK       ((daddr_t)(0))
#define SBLOCK       ((daddr_t)(BBLOCK + BBSIZE / DEV_BSIZE))
#define UFSROOTINO   ((ino_t)2)
#define LOSTFOUNDINO (UFSROOTINO + 1)
#define NRPOS        8
#define MAXIPG       2048
#define MINBSIZE     4096
#define MAXCPG       32
#define MAXMNTLEN    512
#define MAXCSBUFS    32
#define FS_OPTTIME   0
```

Software Distribution Formats **2-5**

DRAFT COPY
March 19, 1997
File: abi_386/chap2 (Delta 44.3)
386:adm.book:sum

Page: 23

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

⚠ **CAUTION** The distribution of software in filesystem format is Level 2 as of January 1,   G
1993.

3 LOW-LEVEL SYSTEM INFORMATION

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

Type	C	`sizeof`	Alignment (bytes)	Intel386 Architecture
Integral	`char` `signed char`	1	1	signed byte
	`unsigned char`	1	1	unsigned byte
	`short` `signed short`	2	2	signed halfword
	`unsigned short`	2	2	unsigned halfword

	`int`
	`signed int`
	`long`
	`signed long`
	`enum`
		4	4	signed word

	`unsigned int` `unsigned long`	4	4	unsigned word
Pointer	*any-type* `*` *any-type* `(*)()`	4	4	unsigned word
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.

<table>
<tr><td>⚠️ CAUTION</td><td>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.</td></tr>
</table>

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

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

Aggregates and Unions

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

- An entire structure or union object is aligned on the same boundary as its most strictly aligned member.

- Each member is assigned to the lowest available offset with the appropriate alignment. This may require *internal padding*, depending on the previous member.

- A structure's size is increased, if necessary, to make it a multiple of the alignment. This may require *tail padding*, depending on the last member.

Machine Interface   **3-3**

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 29

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

Figure 3-2: Structure Smaller Than a Word

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

Byte aligned, `sizeof` is 1

c	0

Figure 3-3: No Padding

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

Word aligned, `sizeof` is 8

s	2	d	1	c	0
n					4
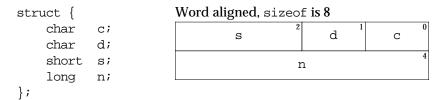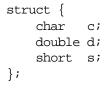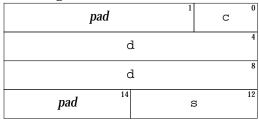
Figure 3-4: Internal Padding

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

Halfword aligned, `sizeof` is 4

s	2	*pad*	1	c	0

Figure 3-5: Internal and Tail Padding

```
struct {                Word aligned, sizeof is 16
    char   c;
    double d;
    short  s;
};
```

pad		c
	1	0

d	4

d	8

pad	14	s	12

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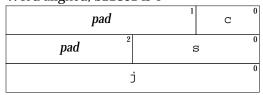
CAUTION A compiler that provides the doubleword alignment mentioned above would arrange the preceding structure differently. Programs built with the double-word alignment facility would not conform to the Intel386 ABI, and they would not be data-compatible with conforming Intel386 programs.

Figure 3-6: union **Allocation**

```
union {                 Word aligned, sizeof is 4
    char   c;
    short  s;
    int    j;
};
```

pad	1	c	0
pad	2	s	0
j			0

Machine Interface   **3-5**

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 31

Bit-Fields

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

Figure 3-7: Bit-Field Ranges

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

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

- Bit-fields are allocated from right to left (least to most significant).

- A bit-field must entirely reside in a storage unit appropriate for its declared type. Thus a bit-field never crosses its unit boundary.

- Bit-fields may share a storage unit with other `struct`/`union` members, including members that are not bit-fields. Of course, `struct` members occupy different parts of the storage unit.

- Unnamed bit-fields' types do not affect the alignment of a structure or union, although individual bit-fields' member offsets obey the alignment constraints.

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

```
0x01020304    | 01    3| 02    2| 03    1| 04    0|
              |31   24|23   16|15    8|7      0|
```

Figure 3-9: Right-to-Left Allocation

```
struct {            Word aligned, sizeof is 4
    int    j:5;     |     pad       | m  | k  | j 0|
    int    k:6;     |31          18|17 11|10 5|4 0|
    int    m:7;
};
```

Figure 3-10: Boundary Alignment

```
struct {            Word aligned, sizeof is 12
    short  s:9;     | c    3| pad  | j   | s  0|
    int    j:9;     |     |23  18|17  9|8   0|
    char   c;       | pad  | u  6| pad  | t  4|
    short  t:9;     |15  9|8  0|15  9|8   0|
    short  u:9;     |     pad         9| d  8|
    char   d;       |
};
```
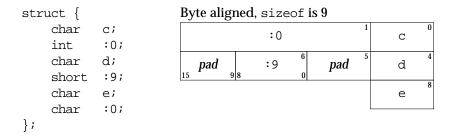
Figure 3-11: Storage Unit Sharing

```
struct {                Halfword aligned, sizeof is 2
    char   c;
    short  s:8;
};
```

1 s	0 c
15 8	

Figure 3-12: `union` **Allocation**

```
union {                 Halfword aligned, sizeof is 2
    char   c;
    short  s:8;
};
```

1 *pad*	0 c
15 8	0 s
pad	
15 8	7 0

Figure 3-13: Unnamed Bit-Fields

```
struct {                Byte aligned, sizeof is 9
    char   c;
    int    :0;
    char   d;
    short  :9;
    char   e;
    char   :0;
};
```

1 :0	0 c
pad :9 *pad*	d
	e

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

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

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

Figure 3-15: Standard Stack Frame

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

Several key points about the stack frame deserve mention.

■ The stack is word aligned. Although the architecture does not require any   G
  alignment of the stack, software convention and the operating system   G
  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.   G

%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.

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

%st(0)   *Floating-point return values* appear on the top of the floating-
                 point register stack; there is no difference in the representation
                 of single- or double-precision values in floating-point registers.
                 If the function does not return a floating-point value, then this
                 register must be empty. This register must be empty before   G
                 entry to a function.

%st(1) through %st(7)
                 Floating-point scratch registers have no specified role in the stan-
                 dard calling sequence. These registers must be empty before
                 entry and upon exit from a function.

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

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

Signals can interrupt processes [see signal(BA_OS)]. Functions called during sig-
nal 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

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

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

Function Calling Sequence   **3-13**

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 39

Figure 3-17: Function Epilogue

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

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

Position	After call	After ret	Position
4*n*+4(%esp)	argument word *n*	argument word *n*	4*n*−4(%esp)
	. . .	. . .	
8(%esp)	argument word 1	argument word 1	0(%esp)
4(%esp)	value address	*undefined*	
0(%esp)	return address		

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

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
```

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.

Figure 3-21: Integral and Pointer Arguments

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

Floating-Point Arguments

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

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 43

Figure 3-22: Floating-Point Arguments

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

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

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

Operating System Interface

Virtual Address Space

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

Page Size

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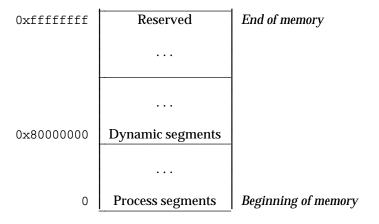
Virtual Address Assignments

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

- The system reserves a configuration-dependent amount of virtual space.

- The system reserves a configuration dependent amount of space per pro-   G
  cess.

- 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 pro-
  gram execution to the next, affects the available amounts.

Figure 3-24: Virtual Address Configuration

```
0xffffffff   │   Reserved    │   End of memory
             │               │
             │     . . .     │
             │               │
             ├───────────────┤
             │               │
             │     . . .     │
             │               │
0x80000000   │ Dynamic segments │
             ├───────────────┤
             │               │
             │     . . .     │
             │               │
       0     │ Process segments │   Beginning of memory
             └───────────────┘
```

CAUTION Programs that dereference null pointers are erroneous. although an imple-   G
mentation is not obliged to detect such erroneous behavior. Such pro-   G
grams 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 infor-
mation 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.

Operating System Interface   **3-21**

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 47
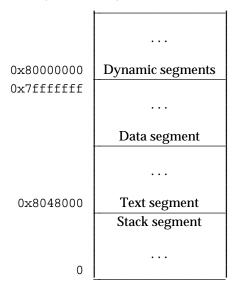
Although applications may control their memory assignments, the typical arrangement appears below.

Figure 3-25: Conventional Segment Arrangements

```
                          |                   |
                          |        . . .      |
                          |                   |
            0x80000000    |  Dynamic segments |
            0x7fffffff    |-------------------|
                          |                   |
                          |        . . .      |
                          |                   |
                          |    Data segment   |
                          |-------------------|
                          |                   |
                          |        . . .      |
                          |                   |
            0x8048000     |    Text segment   |
                          |-------------------|
                          |   Stack segment   |
                          |                   |
                          |        . . .      |
                     0    |                   |
```

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).

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 50

Figure 3-27: Hardware Exceptions and Signals

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

Floating-point instructions exist in the architecture, but they may be implemented
either in hardware (via the Intel387 chip) or in software (via the Intel387 emula-
tor). 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 pro-
cess. 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

```
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

Figure 3-29: EFLAGS Register Fields

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

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

Figure 3-30: Floating-Point Control Word

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

CAUTION The initial floating-point state should be changed with care. In particular, G
many floating-point routines may produce undefined behavior if the preci- G
sion control is set to less than 53 bits. The _fpstart routine (see Chapter 6) G
changes the precision control to 64 bits and sets all exceptions to be G
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

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

`4(%esp)` aligns with "(*Argument count* words)" row
`0(%esp)` aligns with "Argument count" row

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.

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 55

Figure 3-32: Auxiliary Vector

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

Figure 3-33: Auxiliary Vector Types, `a_type`

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

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

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 auxili-
                ary 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 supple-
                ment 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

Figure 3-34: AT_FPHW values

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

AT_INTP_DEVICE

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

AT_INTP_INODE

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

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

To illustrate, suppose an example process receives two arguments.

- echo

- abi

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

- HOME=/home/dir

- PATH=/usr/bin:

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

- {AT_EXECFD, 13}

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

Figure 3-35: Example Process Stack

address	col1	col2	col3	col4	label
	n	:	\0	*pad*	*High addresses*
	r	/	b	i	
	=	/	u	s	
0x8047ff0	P	A	T	H	
	d	i	r	\0	
	o	m	e	/	
	E	=	/	h	
0x8047fe0	\0	H	O	M	
	\0	a	b	i	
	e	c	h	o	
	0				
0x8047fd0	0				
	13				
	2				Auxiliary vector
	0				
0x8047fc0	0x8047ff0				
	0x8047fe1				Environment vector
	0				
	0x8047fdd				
0x8047fb0	0x8047fd8				Argument vector
0(%esp), 0x8047fac	2				Argument count
	Undefined				*Low addresses*

Operating System Interface 3-33

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 59

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

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 calcu-
lates its value once at the entry.

Figure 3-36: Calculating Global Offset Table Address

Line   Code

```
1          call   .L1
2   .L1:   popl   %ebx
3          addl   $_GLOBAL_OFFSET_TABLE_+[.-.L1], %ebx
```

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 dis-
   tance from the addl instruction to the global offset table; [.-.L1] gives the
   distance from .L1 to the addl instruction. Adding their sum to the abso-
   lute address of .L1, already in %ebx, gives the absolute address of the glo-
   bal 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.

Figure 3-38: Absolute Data Access

C	Assembly
| ```
extern int src;
extern int dst;
extern int *ptr;
ptr = &dst;

*ptr = src;
``` | ```
.globl    src, dst, ptr


movl      $dst, ptr

movl      ptr, %eax
movl      src, %edx
movl      %edx, (%eax)
``` |

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

Figure 3-39: Position-Independent Data Access

C	Assembly
| ```
extern int src;
extern int dst;
extern int *ptr;
ptr = &dst;



*ptr = src;
``` | ```
.globl    src, dst, ptr


movl      ptr@GOT(%ebx), %eax
movl      dst@GOT(%ebx), %edx
movl      %edx, (%eax)


movl      ptr@GOT(%ebx), %eax
movl      (%eax), %eax
movl      src@GOT(%ebx), %edx
movl      (%edx), %edx
movl      %edx, (%eax)
``` |

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

Figure 3-40: Position-Independent Static Data Access

C	Assembly
| ```
static int src;
static int dst;
static int *ptr;
ptr = &dst;



*ptr = src;
``` | ```
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 infor-
mation on the indirection sequence.

Figure 3-41: Absolute Direct Function Call

C	Assembly
| ```
extern void function();
function();
``` | ```
.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.

Coding Examples   **3-39**

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 65

Figure 3-42: Position-Independent Direct Function Call

C	Assembly
`extern void function();` `function();`	`.globl    function` `call      function@PLT`

Indirect function calls use the indirect `call` instruction.

Figure 3-43: Absolute Indirect Function Call

C	Assembly

`extern void (*ptr)();`
`extern void name();`
`ptr = name;`
 	
`(*ptr)();`
	`.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	Assembly

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

Page: 66

Branching

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

Figure 3-45: Branch Instruction, All Models

```
          C                          Assembly
label:                    .L01:
        . . .                      . . .
        goto label;                jmp     .L01
```

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 `.Lcase`*i*, `.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.

Figure 3-46: Absolute `switch` **Code**

```
            C                                    Assembly
switch (j)                          cmpl    $3, %eax
{                                   ja      .Ldef
case 0:                             jmp     *.Ltab(,%eax,4)
      . . .                 .Ltab:  .long   .Lcase0
case 2:                             .long   .Ldef
      . . .                         .long   .Lcase2
case 3:                             .long   .Lcase3
      . . .
default:
      . . .
}
```

Figure 3-47: Position-Independent `switch` **Code**

```
        C                                      Assembly
switch (j)                 cmpl    $3, %eax
{                          ja      .Ldef
case 0:                    leal    .Ltab@GOTOFF(%ebx), %edx
      . . .                movl    (%edx,%eax,4), %eax
case 2:                    movl    .Ltab@GOTOFF(%ebx,%eax,4), %eax
      . . .                call    .Ljmp
case 3:            .Ljmp:
      . . .                popl    %ecx
default:                   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.   M
This represents one possible organization of the C stack frame. Usage of `%ebp` as a   M
frame pointer, the exact positions of the callee saved registers, and space for local   M
storage is implementation specific.

Figure 3-48: C Stack Frame

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

A C stack frame doesn't normally change size during execution. The exception is
dynamically allocated stack memory, discussed below. By convention, a function
allocates automatic (local) variables in the middle of its frame and references them
as negative offsets from `%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(BA_LIB)]. This resets the process to a new context in a previous stack frame, automatically discarding the dynamic allocation.

- The process can terminate.

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

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

Figure 3-49: Dynamic Stack Allocation

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

Coding Examples   **3-45**

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 71

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.

DRAFT COPY
March 19, 1997
File: abi_386/chap3 (Delta 44.6)
386:adm.book:sum

Page: 72

4 OBJECT FILES

ELF Header

Machine Information

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

Figure 4-1: Intel386 Identification, e_ident

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

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

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

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

.plt   This section holds the procedure linkage table. See ''Procedure Linkage Table'' in Chapter 5 for more information.

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

word32
31   0

---

word32   This specifies a 32-bit field occupying 4 bytes with arbitrary byte
                alignment. These values use the same byte order as other word
                values in the Intel386 architecture.

0x01020304
3 01	2 02	1 03	0 04
31			0

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 ''Pro-
         gram 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 func-
        tion call to the proper destination. The link editor builds the initial pro-
        cedure linkage table, and the dynamic linker modifies the entries during
        execution. See ''Procedure Linkage Table'' in Chapter 5 for more informa-
        tion.

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.

Figure 4-4: Relocation Types

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

5 PROGRAM LOADING AND DYNAMIC LINKING

Program Loading

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

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

Figure 5-1: Executable File

File Offset	File	Virtual Address
0	ELF header	
	Program header table	
	Other information	
0x100	Text segment	0x8048100
	. . .	
	0x2be00 bytes	0x8073eff
0x2bf00	Data segment	0x8074f00
	. . .	
	0x4e00 bytes	0x8079cff
0x30d00	Other information	
	. . .	

Figure 5-2: Program Header Segments

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

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

- The first text page contains the ELF header, the program header table, and other information.

- The last text page holds a copy of the beginning of data.

- The first data page has a copy of the end of text.

- The last data page may contain file information not relevant to the running process.

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

The end of the data segment requires special handling for uninitialized data, which the system defines to begin with zero values. Thus if a file's last data page includes information not in the logical memory page, the extraneous data must be set to zero, not the unknown contents of the executable file. ''Impurities'' in the other three pages are not logically part of the process image; whether the system expunges them is unspecified. The memory image for this program follows, assuming 4 KB (0x1000) pages.

Figure 5-3: Process Image Segments

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

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

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

Figure 5-4: Example Shared Object Segment Addresses

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

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 shara-bility 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 seg-ments 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 sym-bol 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.

Figure 5-5: Global Offset Table

```
extern Elf32_Addr    _GLOBAL_OFFSET_TABLE_[];
```

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

Function Addresses

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

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

DRAFT COPY
March 19, 1997
File: abi_386/chap5 (Delta 44.3)
386:adm.book:sum

Page: 86

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.

Dynamic Linking **5-9**

DRAFT COPY
March 19, 1997
File: abi_386/chap5 (Delta 44.3)
386:adm.book:sum

Page: 89

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   G

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

        /usr/lib/libc.so.1   G

DRAFT COPY
March 19, 1997
File: abi_386/chap5 (Delta 44.3)
386:adm.book:sum

Page: 90

6 LIBRARIES

Shared Library Names M

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

Figure 6-1: Shared Library Names M

Library Reference Name
libc.so.1
libthread.so.1
libdl.so.1
libnsl.so.1
libX11.so.5.0
libXt.so.5.0
libXext.so.5.0
libXm.so.1.2
libMrm.so.1.2

M
M
M
M
M
M
M
M
M
M
M

Additional Entry Points

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

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

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

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

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

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

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

```
int _xstat(int, char *, struct stat *);
```
            The semantics of this function are identical to those of the
`stat`(BA_OS) function described in the *System V Interface*

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.

M

Figure 6-3: `libc`**, Support Routines**

```
            _fpstart   _ _fpstart   sbrk   _sbrk
```

```
char *sbrk(int incr);
```
This function adds *incr* bytes to the *break value* and changes the allocated space accordingly. *Incr* can be negative, in which case the amount of allocated space is decreased. The break value is the address of the first allocation beyond the end of the data segment. The amount of allocated space increases as the break value increases. Newly allocated space is set to zero. If, however, the same memory space is reallocated to the same process, its contents are undefined. Upon successful completion, `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.

C Library **6-3**

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

Global Data Symbols

The libc library requires that some globel external data objects be defined for the   M
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, how-   M
ever, 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*, con-
forming 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 seg-
           ment 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 pro-
           gram wants ANSI C semantics.

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

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.   M

Reentrancy Considerations   M

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

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

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

```
struct  aiocb {
        int             aio_fildes;
        volatile void*  aio_buf;
        size_t          aio_nbytes;
        off_t           aio_offset;
        int             aio_reqprio;
        struct sigevent aio_sigevent;
        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>`

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

Figure 6-7: `<ctype.h>`

```
#define _U    01
#define _L    02
#define _N    04
#define _S    010
#define _P    020
#define _C    040
#define _B    0100
#define _X    0200

extern unsigned char      __ctype[];

#define isalpha(c)  ((__ctype+1)[c]&(_U|_L))
#define isupper(c)  ((__ctype+1)[c]&_U)
#define islower(c)  ((__ctype+1)[c]&_L)
#define isdigit(c)  ((__ctype+1)[c]&_N)
#define isxdigit(c) ((__ctype+1)[c]&_X)
#define isalnum(c)  ((__ctype+1)[c]&(_U|_L|_N))
#define isspace(c)  ((__ctype+1)[c]&_S)
#define ispunct(c)  ((__ctype+1)[c]&_P)
#define isprint(c)  ((__ctype+1)[c]&(_P|_U|_L|_N|_B))
#define isgraph(c)  ((__ctype+1)[c]&(_P|_U|_L|_N))
#define iscntrl(c)  ((__ctype+1)[c]&_C)
#define isascii(c)  (!((c)&~0177))
#define _toupper(c) ((__ctype+258)[c])
#define _tolower(c) ((__ctype+258)[c])
#define toascii(c)  ((c)&0177)
```

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

M

Figure 6-8: `<dirent.h>`

```
typedef struct {
        int             dd_fd;
        int             dd_loc;
        int             dd_size;
        char            *dd_buf;
} DIR;

struct dirent {
        ino_t           d_ino;
        off_t           d_off;
        unsigned short      d_reclen;
        char            d_name[1];
};
```

Figure 6-9: `<dlfcn.h>` *

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

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

```
#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 e_ident[EI_NIDENT];
        Elf32_Half    e_type;
        Elf32_Half    e_machine;
        Elf32_Word    e_version;
        Elf32_Addr    e_entry;
        Elf32_Off     e_phoff;
        Elf32_Off     e_shoff;
        Elf32_Word    e_flags;
        Elf32_Half    e_ehsize;
        Elf32_Half    e_phentsize;
        Elf32_Half    e_phnum;
        Elf32_Half    e_shentsize;
        Elf32_Half    e_shnum;
        Elf32_Half    e_shstrndx;
} Elf32_Ehdr;

#define ELFMAG0             0x7f
#define ELFMAG1             'E'
#define ELFMAG2             'L'
#define ELFMAG3             'F'
#define ELFMAG              "\177ELF"
#define SELFMAG             4
```

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

```
#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 ELFDATANONE        0
#define ELFDATA2LSB        1
#define ELFDATA2MSB        2
#define ELFDATANUM         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
```

6-10   **LIBRARIES**

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

Page: 101

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

```
#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
```

System Data Interfaces 6-11

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

Page: 102

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

```
typedef struct {
        Elf32_Word    sh_name;
        Elf32_Word    sh_type;
        Elf32_Word    sh_flags;
        Elf32_Addr    sh_addr;
        Elf32_Off     sh_offset;
        Elf32_Word    sh_size;
        Elf32_Word    sh_link;
        Elf32_Word    sh_info;
        Elf32_Word    sh_addralign;
        Elf32_Word    sh_entsize;
} Elf32_Shdr;

#define S
```

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

```
#define SHN_UNDEF        0
#define SHN_LORESERVE    0xff00
#define SHN_ABS          0xfff1
#define SHN_COMMON       0xfff2
#define SHN_HIRESERVE    0xffff
#define SHN_LOPROC       0xff00
#define SHN_HIPROC       0xff1f

typedef struct {
        Elf32_Word         st_name;
        Elf32_Addr         st_value;
        Elf32_Word         st_size;
        unsigned char      st_info;
        unsigned char      st_other;
        Elf32_Half         st_shndx;
} Elf32_Sym;

#define STN_UNDEF   0

#define ELF32_ST_BIND(info)           ((info) >> 4)
#define ELF32_ST_TYPE(info)           ((info) & 0xf)
#define ELF32_ST_INFO(bind,type) (((bind)<<4)+((type)&0xf))

#define STB_LOCAL   0
#define STB_GLOBAL  1
#define STB_WEAK    2
#define STB_NUM     3
#define STB_LOPROC 13
#define STB_HIPROC 15
```

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

```
#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_LOPROC 13
#define STT_HIPROC 15

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

typedef struct {
      Elf32_Addr   r_offset;
      Elf32_Word   r_info;
      Elf32_Sword  r_addend;
} Elf32_Rela;

#define ELF32_R_SYM(info) ((info)>>8)
#define ELF32_R_TYPE(info)((unsigned char)(info))
#define ELF32_R_INFO(sym,type)   (((sym)<<8)+(unsigned char)(type))
```

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

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

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

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

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

```
#define EADV           68
#define ESRMNT         69
#define ECOMM          70
#define EPROTO         71
#define EMULTIHOP      74
#define EBADMSG        77
#define ENAMETOOLONG   78
#define EOVERFLOW      79
#define ENOTUNIQ       80
#define EBADFD         81
#define EREMCHG        82
#define ENOSYS         89
#define ELOOP          90
#define ERESTART       91
#define ESTRPIPE       92
#define ENOTEMPTY      93
#define EUSERS         94
#define ECONNABORTED   130                    G
#define CONNRESET      131                    G
#define ECONNREFUSED              146         G
#define EINPROGRESS    150                    M
#define ESTALE         151
#define ECANCELED      158                    M

/* Non-reentrant */                           M
extern int errno;                             M

/* Reentrant */                               M
#define errno    (*__thr_errno())             M
```

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

```
#define O_RDONLY          0
#define O_WRONLY          1
#define O_RDWR            2
#define O_NDELAY          0x04
#define O_APPEND          0x08
#define O_SYNC            0x10
#define O_NONBLOCK        0x80
#define O_CREAT           0x100
#define O_TRUNC           0x200
#define O_EXCL            0x400
#define O_NOCTTY          0x800

#define F_DUPFD           0
#define F_GETFD           1
#define F_SETFD           2
#define F_GETFL           3
#define F_SETFL           4
#define F_GETLK           14
#define F_SETLK           6
#define F_SETLKW          7
```

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

```
typedef struct flock {
        short  l_type;
        short  l_whence;
        off_t  l_start;
        off_t  l_len;
        long   l_sysid;
        pid_t  l_pid;
        long   pad[4];
} flock_t;

#define F_RDLCK          01
#define F_WRLCK          02
#define F_UNLCK          03

#define O_ACCMODE        3
#define FD_CLOEXEC       1
```

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

```
extern int          __flt_rounds;
#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**

```
#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**

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

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

```
#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**

```
#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: `<fnmatch.h>`*

```
#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>`

```
#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>` *

```
#define GLOB_APPEND        0x0001
#define GLOB_DOOFFS        0x0002
#define GLOB_ERR           0x0004
#define GLOB_MARK          0x0008
#define GLOB_NOCHECK       0x0010
#define GLOB_NOSORT        0x0020
#define GLOB_NOESCAPE      0x0040
#define GLOB_OKAYDOT       0x0200
#define GLOB_BADRANGE      0x0400
#define GLOB_EXTENDED      0x1000

#define GLOB_NOSYS         (-1)
#define GLOB_ABORTED       (-2)
#define GLOB_NOSPACE       (-3)
#define GLOB_NOMATCH       (-4)

typedef struct
{
      void          *;
      char          **gl_pathv;
      size_t        gl_pathc;
      size_t        gl_offs;
} glob_t;
```

Page: 115

Figure 6-29: `<grp.h>`

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

Figure 6-30: `<iconv.h>`*

```
typedef void *iconv_t;
```

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

```
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
```

Page: 117

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

```
#define DAY_1            1
#define DAY_2            2
#define DAY_3            3
#define DAY_4            4
#define DAY_5            5
#define DAY_6            6
#define DAY_7            7

#define ABDAY_1          8
#define ABDAY_2          9
#define ABDAY_3          10
#define ABDAY_4          11
#define ABDAY_5          12
#define ABDAY_6          13
#define ABDAY_7          14

#define MON_1            15
#define MON_2            16
#define MON_3            17
#define MON_4            18
#define MON_5            19
#define MON_6            20
#define MON_7            21
#define MON_8            22
#define MON_9            23
#define MON_10           24
#define MON_11           25
#define MON_12           26
```

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

```
#define ABMON_1          27
#define ABMON_2          28
#define ABMON_3          29
#define ABMON_4          30
#define ABMON_5          31
#define ABMON_6          32
#define ABMON_7          33
#define ABMON_8          34
#define ABMON_9          35
#define ABMON_10         36
#define ABMON_11         37
#define ABMON_12         38

#define RADIXCHAR        39
#define THOUSEP          40
#define YESSTR           41
#define NOSTR            42
#define CRNCYSTR         43

#define D_T_FMT          44
#define D_FMT            45
#define T_FMT            46
#define AM_STR           47
#define PM_STR           48
#define CODESET          49        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
```

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

Page: 119

Figure 6-34: `<limits.h>`**, Part 1 of 2**

```
#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**

```
#define NL_ARGMAX          9
#define NL_LANGMAX         14
#define NL_MSGMAX          32767
#define NL_NMAX            1
#define NL_SETMAX          255
#define NL_TEXTMAX         255
#define NZERO              20

#define WORD_BIT           32
#define LONG_BIT           32

#define DBL_DIG            15
#define DBL_MAX            1.7976931348623157E+308
#define DBL_MIN            2.2250738585072014E-308
#define FLT_DIG            6
#define FLT_MAX            3.40282347E+38F
#define FLT_MIN            1.17549435E-38F

#define FCHR_MAX           1048576
```

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;
};

#define LC_CTYPE            0
#define LC_NUMERIC          1
#define LC_TIME             2
#define LC_COLLATE          3
#define LC_MONETARY         4
#define LC_MESSAGES         5
#define LC_ALL              6
```

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

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

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

Figure 6-38: `<machlock.h>` *

```
typedef volatile unsigned char _simplelock_t;
```

Figure 6-39: `<math.h>`

```
extern const double __huge_val;                          G
#define HUGE_VAL    __huge_val                           G
```

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

```
#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)                      G
#define PROC_DATA    (PROT_READ | PROT_WRITE | PROT_EXEC)         G

#define SHARED             0x10                                   G
#define PRIVATE            0x20                                   G

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

#define      MCL_CURRENT  0x1                                     G
#define      MCL_FUTURE   0x2                                     G
```

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

```
#define VOID            void
#define MAXPATHLEN      1024
#define MODMAXLINKINFOLEN 32

struct modspecific_stat {
      char   mss_linkinfo[MODMAXLINKINFOLEN];
      int    mss_type;
      int    mss_p0[2];
      int    mss_p1[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>`

```
#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>`

```
#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;
};
```

Page: 127

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

G

```
struct netconfig {
        char                *nc_netid;
        unsigned long       nc_semantics;
        unsigned long       nc_flag;
        char                *nc_protofmly;
        char                *nc_proto;
        char                *nc_device;
        unsigned long       nc_nlookups;
        char                **nc_lookups;
        unsigned long       nc_unused[8];
};

#define NC_TPI_CLTS             1
#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**

```
#define NC_NOPROTOFMLY      "-"
#define NC_LOOPBACK         "loopback"
#define NC_INET             "inet"
#define NC_IMPLINK          "implink"
#define NC_PUP              "pup"
#define NC_CHAOS            "chaos"
#define NC_NS               "ns"
#define NC_NBS              "nbs"
#define NC_ECMA             "ecma"
#define NC_DATAKIT          "datakit"
#define NC_CCITT            "ccitt"
#define NC_SNA              "sna"
#define NC_DECNET           "decnet"
#define NC_DLI              "dli"
#define NC_LAT              "lat"
#define NC_HYLINK           "hylink"
#define NC_APPLETALK        "appletalk"
#define NC_NIT              "nit"
#define NC_IEEE802          "ieee802"
#define NC_OSI              "osi"
#define NC_X25              "x25"
#define NC_OSINET           "osinet"
#define NC_GOSIP            "gosip"
#define NC_NETWARE          "netware"           M

#define NC_NOPROTO          "-"
#define NC_TCP              "tcp"
#define NC_UDP              "udp"
#define NC_ICMP             "icmp"
#define NC_IPX              "ipx"               M
#define NC_SPX              "spx"               M
```

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

```
struct nd_addrlist {
      int                n_cnt;
      struct netbuf      *n_addrs;
};

struct nd_hostservlist {
      int                h_cnt;
      struct nd_hostserv *h_hostservs;
};

struct nd_hostserv {
      char         *h_host;
      char         *h_serv;
};

#define ND_HOSTSERV           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_UKNWN              6
#define ND_NOCTRL             7
#define ND_FAILCTRL           8
#define ND_SYSTEM             9
#define ND_NOERRMEM           10
#define ND_NOLIB              11
#define ND_XTIERROR           12
#define ND_BADSTATE           13
```

```
                                                           M
                                                           M
                                                           M
                                                           M
```

System Data Interfaces   6-39

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

Page: 130

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                    M
#define ND_SET_REUSEADDR        6                    M
#define ND_CLEAR_REUSEADDR      7                    M

#define HOST_SELF               "\\1"
#define HOST_ANY                "\\2"
#define HOST_BROADCAST          "\\3"
```

Figure 6-48: `<nl_types.h>`

```
#define NL_SETD         1

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

6-40   **LIBRARIES**

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

Page: 131

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>`

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

#define POLLIN          0x0001
#define POLLPRI         0x0002
#define POLLOUT         0x0004
#define POLLRDNORM      0x0040
#define POLLWRNORM      POLLOUT
#define POLLRDBAND      0x0080
#define POLLWRBAND      0x0100


#define POLLNORM        POLLRDNORM


#define POLLERR         0x0008
#define POLLHUP         0x0010
#define POLLNVAL        0x0020
```

6-42   **LIBRARIES**

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

Page: 133

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

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


#define PC_CLNULL          -1


#define PC_CLNMSZ   16
#define PC_CLINFOSZ (32 / sizeof(long))
#define PC_CLPARMSZ (32 / sizeof(long))

typedef struct pcinfo {
        id_t   pc_cid;
        char   pc_clname[PC_CLNMSZ];
        long   pc_clinfo[PC_CLINFOSZ];
} pcinfo_t;

typedef struct pcparms {
        id_t   pc_cid;
        long   pc_clparms[PC_CLPARMSZ];
} pcparms_t;
```

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

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

```
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;
};
```

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

```
#define REG_NOTBOL          0x000001
#define REG_NOTEOL          0x000002
#define REG_NONEMPTY        0x000004

#define REG_OR              0x000001
#define REG_PLUS            0x000002
#define REG_QUEST           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 | REG_QUEST |
              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 REG_ANGLES          0x040000
#define REG_ESCNL           0x080000
#define REG_OLDBRE  (REG_BADRANGE | REG_ANGLES | REG_ESCNL)
```

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

```
#define REG_ENOSYS          (-1)
#define REG_NOMATCH         1
#define REG_BADPAT          2
#define REG_ECOLLATE        3
#define REG_ECTYPE          4
#define REG_EESCAPE         7
#define REG_ESUBREG         8
#define REG_EBRACK          9
#define REG_NOPAT           12
#define REG_EPAREN          13
#define REG_EBRACE          14
#define REG_BADBR           15
#define REG_ERANGE          16
#define REG_ESPACE          17
#define REG_BADRPT          18


typedef struct
{
      size_t         re_nsub;
      unsigned long       re_flags;
      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>`

```
#define RLIMIT_CPU        0
#define RLIMIT_FSIZE      1
#define RLIMIT_DATA       2
#define RLIMIT_STACK      3
#define RLIMIT_CORE       4
#define RLIMIT_NOFILE     5
#define RLIMIT_VMEM       6
#define RLIM_NLIMITS      7
#define RLIMIT_AS         RLIMIT_VMEM
#define RLIM_INFINITY     0x7fffffff


typedef unsigned long     rlim_t;


struct rlimit {
        rlim_t        rlim_cur;
        rlim_t        rlim_max;
};
```

Figure 6-57: `<rpc.h>`**, Part 1 of 16**

```
#define bool_t              int
#define enum_t              int

enum xdr_op {
        XDR_ENCODE=0,
        XDR_DECODE=1,
        XDR_FREE=2
};

typedef bool_t      (*xdrproc_t)();

typedef struct {
        enum xdr_op  x_op;
        struct xdr_ops {
                bool_t (*x_getlong)();
                bool_t (*x_putlong)();
                bool_t (*x_getbytes)();
                bool_t (*x_putbytes)();
                u_int  (*x_getpostn)();
                bool_t (*x_setpostn)();
                long * (*x_inline)();
                void   (*x_destroy)();
        } *x_ops;
        caddr_t       x_public;
        caddr_t       x_private;
        caddr_t       x_base;
        int           x_handy;
} XDR;
```

Figure 6-58: `<rpc.h>`**, Part 2 of 16**

```
#define xdr_getpos(xdrs)                              \
       (*(xdrs)->x_ops->x_getpostn)(xdrs)
#define xdr_setpos(xdrs, pos)                         \
       (*(xdrs)->x_ops->x_setpostn)(xdrs, pos)
#define xdr_inline(xdrs, len)                         \
       (*(xdrs)->x_ops->x_inline)(xdrs, len)
#define xdr_destroy(xdrs)                             \        M
       (*(xdrs)->x_ops->x_destroy)(xdrs)                       M

#define NULL_xdrproc_t     ((xdrproc_t)0)
struct xdr_discrim {
       int          value;
       xdrproc_t    proc;
};
```

Figure 6-59: `<rpc.h>`**, Part 3 of 16**

```
#define MAX_AUTH_BYTES          400
#define MAXNETNAMELEN           255
#define HEXKEYBYTES             48

enum auth_stat {
        AUTH_OK=0,
        AUTH_BADCRED=1,
        AUTH_REJECTEDCRED=2,
        AUTH_BADVERF=3,
        AUTH_REJECTEDVERF=4,
        AUTH_TOOWEAK=5,
        AUTH_INVALIDRESP=6,
        AUTH_FAILED=7
};

typedef u_long      u_int32                                      M

union des_block {
        struct {
                u_int32         high;                           M
                u_int32         low;                            M
        } key;
        char   c[8];
};
typedef union des_block         des_block;
```

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

Page: 142

Figure 6-60: `<rpc.h>`**, Part 4 of 16**

```
struct opaque_auth {
        enum_t          oa_flavor;
        caddr_t         oa_base;
        u_int           oa_length;
};

typedef struct {
        struct opaque_auth  ah_cred;
        struct opaque_auth  ah_verf;
        union des_block     ah_key;
        struct auth_ops {
                void  (*ah_nextverf)();
                int   (*ah_marshal)();
                int   (*ah_validate)();
                int   (*ah_refresh)();
                void  (*ah_destroy)();
        } *ah_ops;
        caddr_t         ah_private;
} AUTH;

#define auth_destroy(auth)          \
            ((*((auth)->ah_ops->ah_destroy))(auth))

#define AUTH_NONE           0
#define AUTH_NULL           0
#define AUTH_SYS            1
#define AUTH_UNIX           AUTH_SYS
#define AUTH_SHORT          2
#define AUTH_DES            3
#define AUTH_ESV            200004
```

M

Figure 6-61: `<rpc.h>`**, Part 5 of 16**

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

Figure 6-62: `<rpc.h>`**, Part 6 of 16**

```
struct rpc_err {
      enum clnt_stat        re_status;
      union {
            struct {
                  int    errno;
                  int    t_errno;
            } RE_err;
            enum auth_stat      RE_why;
            struct {
                  u_long        low;
                  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**

```
#define FEEDBACK_REXMIT1    1
#define FEEDBACK_OK         2

#define clnt_call(rh, proc, xargs, argsp, xres, resp, secs)\
       ((*(rh)->cl_ops->cl_call)                           \
       (rh, proc, xargs, argsp, xres, resp, secs))
#define clnt_abort(rh)                                     \
       ((*(rh)->cl_ops->cl_abort)(rh))
#define clnt_geterr(rh, errp)                              \
       ((*(rh)->cl_ops->cl_geterr)(rh, errp))
#define clnt_freeres(rh,xres,resp)                         \
       ((*(rh)->cl_ops->cl_freeres)(rh,xres,resp))
#define clnt_control(cl, rq, in)                           \
       ((*(cl)->cl_ops->cl_control)(cl, rq, in))
#define clnt_destroy(rh)                                   \
       ((*(rh)->cl_ops->cl_destroy)(rh))

#define CLSET_TIMEOUT              1
#define CLGET_TIMEOUT              2
#define CLGET_SERVER_ADDR         3
#define CLGET_FD                  6
#define CLGET_SVC_ADDR            7
#define CLSET_FD_CLOSE            8
#define CLSET_FD_NCLOSE           9
#define CLSET_RETRY_TIMEOUT       4
#define CLGET_RETRY_TIMEOUT       5
```

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

Page: 146

Figure 6-64: `<rpc.h>`**, Part 8 of 16**

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

extern  rpc_createerr_t         rpc_createerr;        M
```

Figure 6-65: `<rpc.h>`**, Part 9 of 16**

```
enum msg_type {
        CALL=0,
        REPLY=1
};

enum reply_stat {
        MSG_ACCEPTED=0,
        MSG_DENIED=1
};

enum accept_stat {
        SUCCESS=0,
        PROG_UNAVAIL=1,
        PROG_MISMATCH=2,
        PROC_UNAVAIL=3,
        GARBAGE_ARGS=4,
        SYSTEM_ERR=5
};

enum reject_stat {
        RPC_MISMATCH=0,
        AUTH_ERROR=1
};
```

Figure 6-66: `<rpc.h>`**, Part 10 of 16**

```
struct accepted_reply {
        struct opaque_auth          ar_verf;
        enum accept_stat            ar_stat;
        union {
                struct {
                        u_long        low;
                        u_long        high;
                } AR_versions;
                struct {
                        caddr_t       where;
                        xdrproc_t     proc;
                } AR_results;

        } ru;
};

struct rejected_reply {
        enum reject_stat    rj_stat;
        union {
                struct {
                        u_long        low;
                        u_long        high;
                } RJ_versions;
                enum auth_stat      RJ_why;
        } ru;
};
```

Figure 6-67: `<rpc.h>`**, Part 11 of 16**

```
struct reply_body {
        enum reply_stat     rp_stat;
        union {
                struct accepted_reply RP_ar;
                struct rejected_reply RP_dr;
        } ru;
};

struct call_body {
        u_long                  cb_rpcvers;
        u_long                  cb_prog;
        u_long                  cb_vers;
        u_long                  cb_proc;
        struct opaque_auth  cb_cred;
        struct opaque_auth  cb_verf;
};

struct rpc_msg {
        u_long                  rm_xid;
        enum msg_type        rm_direction;
        union {
                struct call_body    RM_cmb;
                struct reply_body   RM_rmb;
        } ru;
};
```

Figure 6-68: `<rpc.h>`**, Part 12 of 16**

```
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**

```
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                    adc_nickname;
};
```

Figure 6-70: `<rpc.h>`**, Part 14 of 16**

```
enum xprt_stat {
        XPRT_DIED,
        XPRT_MOREREQS,
        XPRT_IDLE
};

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

M

System Data Interfaces 6-61

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

Page: 152

Figure 6-71: `<rpc.h>`**, Part 15 of 16**

```
#define svc_getrpccaller(x) (&(x)->xp_rtaddr)
#define svc_getargs(xprt, xargs, argsp)              \
        (*(xprt)->xp_ops->xp_getargs)((xprt), (xargs), (argsp))
#define svc_freeargs(xprt, xargs, argsp)             \
        (*(xprt)->xp_ops->xp_freeargs)((xprt), (xargs), (argsp))
#define svc_destroy(xprt)                            \
        (*(xprt)->xp_ops->xp_destroy)(xprt)

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


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

extern fd_set svc_fdset;
```

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

Figure 6-72: `<rpc.h>`**, Part 16 of 16**

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

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

Figure 6-73: `<rtpriocntl.h>`*

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

typedef struct rtinfo {
        short  rt_maxpri;
} rtinfo_t;

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

Page: 154

Figure 6-74: `<search.h>`

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

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

```
#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>`

```
#define _SIGJBLEN          128
#define _JBLEN             10

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

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

```
#define SHMLBA              ((1)<<12)

#define SHM_RDONLY          010000
#define SHM_RND             020000

struct shmid_ds {
        struct ipc_perm    shm_perm;
        int            shm_segsz;
        struct anon_map    *shm_amp;
        ushort         shm_lkcnt;
        pid_t          shm_lpid;
        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**

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

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

```
#define SIGXCPU                 30
#define SIGXFSZ                 31

#define SIG_DFL         (void(*)())0
#define SIG_ERR         (void(*)())-1
#define SIG_IGN         (void(*)())1
#define SIG_HOLD        (void(*)())2

#define SIG_BLOCK       1
#define SIG_UNBLOCK     2
#define SIG_SETMASK     3

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

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_NOCLDWAIT     0x00010000
```

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

```
#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**

```
#define  ILL_ILLOPC        1
#define  ILL_ILLOPN        2
#define  ILL_ILLADR        3
#define  ILL_ILLTRP        4
#define  ILL_PRVOPC        5
#define  ILL_PRVREG        6
#define  ILL_COPROC        7
#define  ILL_BADSTK        8

#define  FPE_INTDIV        1
#define  FPE_INTOVF        2
#define  FPE_FLTDIV        3
#define  FPE_FLTOVF        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**

```
#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)
```

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

```
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;
```

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

Page: 162

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

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

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

```
union sigval {
        int     sival_int;
        void    *sival_ptr;
};

union notifyinfo {
        int             nisigno;
        void            (*nifunc)(union sigval);
};

struct sigevent {
        int             sigev_notify;
        union notifyinfo sigev_notifyinfo;
        union sigval    sigev_value;
};

#define SIGEV_NONE      1
#define SIGEV_SIGNAL    2
#define SIGEV_CALLBACK  3
```

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

```
#define _ST_FSTYPSZ        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;
        timestruc_t   st_atim;
        timestruc_t   st_mtim;
        timestruc_t   st_ctim;
        long          st_blksize;
        long          st_blocks;
        char          st_fstype[_ST_FSTYPSZ];
        long          st_pad4[8];
};
#define st_atime    st_atim.tv_sec
#define st_mtime    st_mtim.tv_sec
#define st_ctime    st_ctim.tv_sec
```

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

```
#define S_IFMT              0xF000
#define S_IFIFO             0x1000
#define S_IFCHR             0x2000
#define S_IFDIR             0x4000
#define S_IFBLK             0x6000
#define S_IFREG             0x8000
#define S_IFLNK             0xA000

#define S_ISUID             0x800
#define S_ISGID             0x400
#define S_ISVTX             0x200

#define 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>`

```
#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>`

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

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

```
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 P_tmpdir         "/var/tmp/"
#define L_tmpnam         25

#define stdin            (&__iob[0])
#define stdout           (&__iob[1])
#define stderr           (&__iob[2])
```

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

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

extern FILE        __iob[];                                            G

#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  †

/* Non reentrant */                                                   M
#define getc_unlocked(p)      (--(p)->_cnt < 0 ? _ _filbuf(p)         M
             : (int)*(p)->_ptr++)                                     M
#define putc_unlocked(x, p)   (--(p)->_cnt < 0 ? _ _flsbuf(x, p)      M
             : (int)(*(p)->_ptr++ = (x)))                             M
#define getchar_unlocked()    getc_unlocked(stdin)                    M
#define putchar_unlocked(x)   putc_unlocked((x), stdout)              M

#define getc(p)     (--(p)->_cnt < 0 ? _ _filbuf(p)                   M
             : (int)*(p)->_ptr++)                                     M
#define putc(x, p)  (--(p)->_cnt < 0 ? _ _flsbuf(x, p)               M
             : (int)(*(p)->_ptr++ = (x)))                             M
#define getchar()   getc(stdin)                                       M
#define putchar(x)  putc((x), stdout)                                 M

/* Reentrant versions available as functions only */                 M
```

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

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

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

Page: 169

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

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

Figure 6-93: `<stdlib.h>`

```
typedef struct {
        int    quot;
        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]
```

System Data Interfaces 6-79

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

Page: 170

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

```
#define SNDZERO          0x001
#define SNDPIPE          0x002

#define RNORM            0x000
#define RMSGD            0x001
#define RMSGN            0x002
#define RMODEMASK        0x003
#define RPROTDAT         0x004
#define RPROTDIS         0x008
#define RPROTNORM        0x010

#define FLUSHR           0x01
#define FLUSHW           0x02
#define FLUSHRW          0x03
#define FLUSHBAND        0x04
```

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

```
#define S_INPUT          0x0001
#define S_HIPRI          0x0002
#define S_OUTPUT         0x0004
#define S_MSG            0x0008
#define S_ERROR          0x0010
#define S_HANGUP         0x0020
#define S_RDNORM         0x0040
#define S_WRNORM         S_OUTPUT
#define S_RDBAND         0x0080
#define S_WRBAND         0x0100
#define S_BANDURG        0x0200

#define RS_HIPRI         0x01

#define MSG_HIPRI        0x01
#define MSG_ANY          0x02
#define MSG_BAND         0x04

#define MORECTL          1
#define MOREDATA         2

#define MUXID_ALL        (-1)

#define ANYMARK          0x01
#define LASTMARK         0x02
```

System Data Interfaces 6-81

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

Page: 172

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

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

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

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

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

```
struct strioctl {
        int    ic_cmd;
        int    ic_timout;
        int    ic_len;
        char   *ic_dp;
};

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

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

struct strfdinsert {
        struct strbuf      ctlbuf;
        struct strbuf      databuf;
        long               flags;
        int                fildes;
        int                offset;
};
```

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

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

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

#define FMNAMESZ    8

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

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

struct bandinfo {
      unsigned char       bi_pri;
      int                 bi_flag;
};
```

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

Page: 176

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

```
#define USYNC_THREAD    0
#define USYNC_PROCESS   1

typedef struct thrq_elt thrq_elt_t;

struct thrq_elt {
      thrq_elt_t *thrq_next;
      thrq_elt_t *thrq_prev;
 };

typedef volatile struct {
      lwp_mutex_t   m_lmutex;
      long          m_type;
      lwp_mutex_t   m_sync_lock;
      thrq_elt_t    m_sleepq;
      long          filler;
} mutex_t;

typedef volatile struct {
      lwp_cond_t    c_lcond;
      long          c_type;
      thrq_elt_t    *c_syncq;
      lwp_mutex_t   c_sync_lock;
} cond_t;
```

6-86   **LIBRARIES**

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

Page: 177

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

```
typedef volatile struct {
        mutex_t        s_mutex;
        cond_t         s_cond;
        short          s_count;
        short          s_wakecnt;
        int            s_type;
} sema_t;

typedef volatile struct rwcv rwcv_t;

struct rwcv {
        cond_t rwcv_cond;
        rwcv_t *rwcv_next;
        char   rwcv_rw;
        char   rwcv_wakeup;
        short  rwcv_readerwanted;
} ;

typedef volatile struct rwlock rwlock_t;

struct rwlock {
        mutex_t        rw_mutex;
        lwp_cond_t     rw_lwpcond;
        int            rw_type;
        short          rw_readers;
        char           rw_writer;
        char           rw_wrwakeup;
        short          rw_writerwanted;
        short          rw_rdwakecnt;
        rwcv_t         *rw_cvqhead;
        rwcv_t         *rw_cvqtail;
        long           pad[4];
} ;
```

System Data Interfaces 6-87

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

Page: 178

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

```
typedef volatile struct {
        mutex_t        rm_mutex;
        pid_t          rm_pid;
        thread_t       rm_owner;
        int            rm_depth;
        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>`

```
#define SI86FPHW          40

#define FP_NO             0
#define FP_SW             1
#define FP_HW             2
#define FP_287            2
#define FP_387            3
```

6-88 **LIBRARIES**

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

Page: 179

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

```
#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];
};
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 TIOCSWINSZ        (TIOC|103)

#define TCGETS            (TIOC|13)
#define TCSETS            (TIOC|14)
#define TCSANOW           (('T'<<8)|14)
#define TCSETSW           (TIOC|15)
#define TCSADRAIN         (('T'<<8)|15)
#define TCSETSF           (TIOC|16)
#define TCSAFLUSH         (('T'<<8)|16)
```

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

Page: 188

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

```
#define TCIFLUSH          0
#define TCOFLUSH          1
#define TCIOFLUSH         2

#define TCOOFF            0
#define TCOON             1
#define TCIOFF            2
#define TCION             3

#define B0                0
#define B50               1
#define B75               2
#define B110              3
#define B134              4
#define B150              5
#define B200              6
#define B300              7
#define B600              8
#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

```
#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

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

struct ts_param {
        int     prio;
};

struct fifo_param {
        int     prio;
};

struct rr_param {
        int     prio;
};

typedef unsigned int thread_key_t;
```

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

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

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

```
#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>`

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

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

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

LIBRARIES

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

Page: 193

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

```
typedef long clock_t;
typedef long time_t;
typedef unsigned int size_t;                             M

typedef struct timespec {                                M
        time_t          tv_sec;                          M
        long            tv_nsec;                         M
} timestruc_t;                                           M

#define CLOCKS_PER_SEC          1000000

struct tm {
        int     tm_sec;
        int     tm_min;
        int     tm_hour;
        int     tm_mday;
        int     tm_mon;
        int     tm_year;
        int     tm_wday;
        int     tm_yday;
        int     tm_isdst;
};

extern char *tzname[2];

#define CLK_TCK     _sysconf(3)

extern long timezone;
extern int daylight;
```

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   M
as the current tiuser.h. The new xti.h header is a superset of the previous   M
edition's tiuser.h.   M |

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 TSYSERR         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           M
#define TBADNAME        21           M
#define TBADQLEN        22           M
#define TADDRBUSY       23           M
#define TINDOUT         24           M
#define TPROVMISMATCH   25           M
#define TRESQLEN        26           M
#define TRESADDR        27           M
#define TQFULL          28           M
#define TPROTO          29           M
```

System Data Interfaces **6-105**

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

Page: 196

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        M
#define T_GOEXDATA        0x0200        M
#define T_EVENTS          0x03ff        M
```

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

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

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        0x0000001
```
                                                                    M

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

```
struct t_info {
        long    addr;
        long    options;
        long    tsdu;
        long    etsdu;
        long    connect;
        long    discon;
        long    servtype;
        long    flags;                                  M
};

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

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

Page: 199

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

```
struct t_discon {
        struct netbuf       udata;
        int                 reason;
        int                 sequence;
};

struct t_call {
        struct netbuf       addr;
        struct netbuf       opt;
        struct netbuf       udata;
        int                 sequence;
};

struct t_unitdata {
        struct netbuf       addr;
        struct netbuf       opt;
        struct netbuf       udata;
};

struct t_uderr {
        struct netbuf       addr;
        struct netbuf       opt;
        long                error;
};
```

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

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

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

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

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

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

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

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

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

Figure 6-133: `<tspriocntl.h>`*

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


typedef struct tsinfo {
        short  ts_maxupri;
} tsinfo_t;

#define       TS_NOCHANGE   -32768
```

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

```
typedef unsigned char      uchar_t;
typedef unsigned short     ushort_t;
typedef unsigned int       uint_t;
typedef unsigned long      ulong_t;

typedef 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
```

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

```
typedef struct fpregset {
      union {
              struct fpchip_state {
                      int    state[27];
                      int    status;
              } fpchip_state;
              struct fp_emul_space {
                      char   fp_emul[246];
                      char   fp_epad[2];
              } fp_emul_space;
              int    f_fpregs[62];
      } fp_reg_set;
      long   f_wregs[33];
} fpregset_t;

typedef struct {
      gregset_t     gregs;
      fpregset_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>

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

Figure 6-138: <ulimit.h>

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

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

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

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

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

#define _SC_ARG_MAX             1
#define _SC_CHILD_MAX           2
#define _SC_CLK_TCK             3
```

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

```
#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
```

6-118   **LIBRARIES**

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

Page: 209

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

```
#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>

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

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

```
#define SYS_NMLN    257

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

Figure 6-143: `<wait.h>`

```
#define WEXITED           0001
#define WTRAPPED          0002
#define WSTOPPED          0004
#define WCONTINUED        0010
#define WUNTRACED         0004
#define WNOHANG           0100
#define WNOWAIT           0200

#define WCONTFLG          0177777
#define WCOREFLG          0200

#define WWORD(stat)       ((int)((stat))&0177777)

#define WSTOPFLG          0177
#define WSIGMASK          0177
#define WLOBYTE(stat) ((int)((stat)&0377))
#define WHIBYTE(stat) ((int)(((stat)>>8)&0377))

#define WIFEXITED(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)
```

Figure 6-144: <wchar.h>

```
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)   mbrtowc((wchar_t *)0, x, n, p)
```

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

```
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
```

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

```
#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

```
#define iscodeset0(c)      (((c) & ~(wchar_t)0xff) == 0)
#define iscodeset1(c)      (((c) >> 28) == 0x3)
#define iscodeset2(c)      (((c) >> 28) == 0x1)
#define iscodeset3(c)      (((c) >> 28) == 0x2)

inline int __isw(wint_t c, wctype_t t){
      if (c > 255)
            return (__iswctype(c, t));
      return (1 + __ctype)[c] & t ;
}

inline int __isx(wint_t c, wctype_t t){
      return (c > 255 && __iswctype(c, t));
}

inline wint_t __tow(wint_t c, wctype_t t){
      if (c > 255)
            return (__trwctype(c, t));
      if ((1 + __ctype)[c] & t )
            return (258 + __ctype)[c];
      return (c);
}
```

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

System Data Interfaces 6-125

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

Page: 216

Figure 6-148: `<wordexp.h>` *

```
#define WRDE_APPEND      0001              M
#define WRDE_DOOFFS      0002              M
#define WRDE_NOCMD       0004              M
#define WRDE_REUSE       0010              M
#define WRDE_SHOWERR     0020              M
#define WRDE_UNDEF       0040              M


#define WRDE_NOSYS       (-1)             M
#define WRDE_BADCHAR     (-2)             M
#define WRDE_BADVAL      (-3)             M
#define WRDE_CMDSUB      (-4)             M
#define WRDE_NOSPACE     (-5)             M
#define WRDE_SYNTAX      (-6)             M


typedef struct                            M
{                                         M
     size_t we_wordc;                     M
     char   **we_wordv;                   M
     size_t we_offs;                      M
} wordexp_t;                              M
```

6-126   **LIBRARIES**

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

Page: 217

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.

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)
```

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

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

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

Page: 220

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>`

```
extern WidgetClass compositeWidgetClass;
```

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

```
extern WidgetClass constraintWidgetClass;
```

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

```
extern WidgetClass coreWidgetClass;
```

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

```
#define XC_num_glyphs           154
#define XC_X_cursor             0
#define XC_arrow                2
#define XC_based_arrow_down     4
#define XC_based_arrow_up       6
#define XC_boat                 8
#define XC_bogosity             10
#define XC_bottom_left_corner   12
#defind
```

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
```

System Data Interfaces **6-133**

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

Page: 224

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

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

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

```
typedef char *String;

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

typedef void            *Widget;
typedef Widget          *WidgetList;
typedef void            *CompositeWidget;
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            XtArgVal;
typedef unsigned char   XtEnum;

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

typedef void            *XtPointer;
```

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

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

#define XtCWQueryOnly      (1 << 7)
#define XtSMDontChange     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;
```

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

Page: 227

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

```
#define XtInputNoneMask          0L
#define XtInputReadMask          (1L<<0)
#define XtInputWriteMask         (1L<<1)
#define XtInputExceptMask        (1L<<2)

typedef struct {
        XtGeometryMask request_mode;
        Position     x, y;
        Dimension    width, height, border_width;
        Widget sibling;
} XtWidgetGeometry;

typedef struct {
        String       name;
        XtArgVal     value;
} Arg, *ArgList;

typedef XtPointer  XtVarArgsList;

typedef struct {
        XtCallbackProc      callback;
        XtPointer           closure;
} XtCallbackRec, *XtCallbackList;

typedef enum {
        XtCallbackNoList,
        XtCallbackHasNone,
        XtCallbackHasSome
} XtCallbackStatus;

typedef struct {
        Widget       shell_widget;
        Widget       enable_widget;
} XtPopdownIDRec, *XtPopdownID;
```

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

```
typedef enum {
        XtGeometryYes,
        XtGeometryNo,
        XtGeometryAlmost,
        XtGeometryDone
} XtGeometryResult;

typedef enum {
        XtGrabNone,
        XtGrabNonexclusive,
        XtGrabExclusive
} XtGrabKind;

typedef struct {
        String        resource_name;
        String        resource_class;
        String        resource_type;
        Cardinal      resource_size;
        Cardinal      resource_offset;
        String        default_type;
        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[];
```

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

```
#define XtAllEvents   ((EventMask) -1L)
#define XtIMXEvent              1
#define XtIMTimer               2
#define XtIMAlternateInput      4
#define XtIMAll (XtIMXEvent | XtIMTimer | XtIMAlternateInput)

#define XtOffsetOf(s_type,field) XtOffset(s_type*,field)
#define XtNew(type) ((type *) XtMalloc((unsigned) sizeof(type)))
#define XT_CONVERT_FAIL         (Atom)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,(WidgetClass)overrideShellWidgetClass,\
       (WidgetClass)shellWidgetClass, (XtEnum)0x20))
#define XtIsWMShell(widget) \
       (_XtCheckSubclassFlag(widget,(XtEnum)0x40))
#define XtIsVendorShell(widget)      \
       (_XtIsSubclassOf(widget,(WidgetClass)vendorShellWidgetClass,\
       (WidgetClass)wmShellWidgetClass,(XtEnum)0x40))
#define XtIsTransientShell(widget) \
       (_XtIsSubclassOf(widget,(WidgetClass)transientShellWidgetClass,\
       (WidgetClass)wmShellWidgetClass, (XtEnum)0x40))
#define XtIsTopLevelShell(widget)\
       (_XtCheckSubclassFlag(widget, (XtEnum)0x80))
#define XtIsApplicationShell(widget)\
       (_XtIsSubclassOf(widget,(WidgetClass)applicationShellWidgetClass,\
       (WidgetClass)topLevelShellWidgetClass, (XtEnum)0x80))
```

System Data Interfaces 6-139

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

Page: 230

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

```
#define XtSetArg(arg,n,d)\
        ((void)( (arg).name = (n), (arg).value = (XtArgVal)(d) ))
#define XtOffset(p_type,field)\
        ((Cardinal) (((char *) (&(((p_type)NULL)->field))) - ((char *) NULL)))

#define XtVaNestedList              "XtVaNestedList"
#define XtVaTypedArg        "XtVaTypedArg"
#define XtUnspecifiedPixmap  ((Pixmap)2)
#define XtUnspecifiedShellInt       (-1)
#define XtUnspecifiedWindow  ((Window)2)
#define XtUnspecifiedWindowGroup ((Window)3)
#define XtDefaultForeground "XtDefaultForeground"
#define XtDefaultBackground "XtDefaultBackground"
#define XtDefaultFont       "XtDefaultFont"
#define XtDefaultFontSet            "XtDefaultFontSet"
```

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

```
extern WidgetClass objectClass;
```

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

```
extern WidgetClass rectObjClass;
```

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;
```

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

```
typedef unsigned long XID;

typedef XID Window;
typedef XID Drawable;
typedef XID Font;
typedef XID Pixmap;
typedef XID Cursor;
typedef XID Colormap;
typedef XID GContext;
typedef XID KeySym;

typedef unsigned long Atom;
typedef unsigned long VisualID;
typedef unsigned long Time;
typedef unsigned char KeyCode;

#define AllTemporary      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
```

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

Page: 233

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

```
#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)
```

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

```
#define KeyPres
```

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

```
#define ShiftMask          (1<<0)
#define LockMask           (1<<1)
#define ControlMask        (1<<2)
#define Mod1Mask           (1<<3)
#define Mod2Mask           (1<<4)
#define Mod3Mask           (1<<5)
#define Mod4Mask           (1<<6)
#define Mod5Mask           (1<<7)

#define Button1Mask        (1<<8)
#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
```

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

```
#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**

```
#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 CWBitGravity      (1L<<4)
#define CWWinGravity      (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 CWDontPropagate   (1L<<12)
#define CWColormap        (1L<<13)
#define CWCursor          (1L<<14)
```

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

```
#define CWX                (1<<0)
#define CWY                (1<<1)
#define CWWidth            (1<<2)
#define CWHeight           (1<<3)
#define CWBorderWidth      (1<<4)
#define CWSibling          (1<<5)
#define CWStackMode        (1<<6)

#define ForgetGravity          0
#define NorthWestGravity       1
#define NorthGravity           2
#define NorthEastGravity       3
#define WestGravity            4
#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
```

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

Page: 239

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

```
#define Above               0
#define Below               1
#define TopIf               2
#define BottomIf            3
#define Opposite            4
#define RaiseLowest         0
#define LowerHighest        1
#define PropModeReplace     0
#define PropModePrepend     1
#define PropModeAppend      2

#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
```

System Data Interfaces 6-149

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

Page: 240

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

```
#define JoinMiter          0
#define JoinRound          1
#define JoinBevel          2

#define FillSolid          0
#define FillTiled          1
#define FillStippled       2
#define FillOpaqueStippled 3

#define EvenOddRule        0
#define WindingRule        1

#define ClipByChildren     0
#define IncludeInferiors   1

#define Unsorted           0
#define YSorted            1
#define YXSorted           2
#define YXBanded           3

#define CoordModeOrigin    0
#define CoordModePrevious  1

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

#define ArcChord           0
#define ArcPieSlice        1
```

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

```
#define GCFunction              (1L<<0)
#define GCPlaneMask             (1L<<1)
#define GCForeground            (1L<<2)
#define GCBackground            (1L<<3)
#define GCLineWidth             (1L<<4)
#define GCLineStyle             (1L<<5)
#define GCCapStyle              (1L<<6)
#define GCJoinStyle             (1L<<7)
#define GCFillStyle             (1L<<8)
#define GCFillRule              (1L<<9)
#define GCTile                  (1L<<10)
#define GCStipple               (1L<<11)
#define GCTileStipXOrigin       (1L<<12)
#define GCTileStipYOrigin       (1L<<13)
#define GCFont                  (1L<<14)
#define GCSubwindowMode         (1L<<15)
#define GCGraphicsExposures     (1L<<16)
#define GCClipXOrigin           (1L<<17)
#define GCClipYOrigin           (1L<<18)
#define GCClipMask              (1L<<19)
#define GCDashOffset            (1L<<20)
#define GCDashList              (1L<<21)
#define GCArcMode               (1L<<22)

#define FontLeftToRight         0
#define FontRightToLeft         1

#define XYBitmap                0
#define XYPixmap                1
#define ZPixmap                 2

#define AllocNone               0
#define AllocAll                1

#define DoRed                   (1<<0)
#define DoGreen                 (1<<1)
#define DoBlue                  (1<<2)
```

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

Page: 242

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

```
#define CursorShape             0
#define TileShape               1
#define StippleShape            2

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

#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
```

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

```
#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
```

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

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

#define XcmsUndefinedFormat     (XcmsColorFormat)0x00000000
#define XcmsCIEXYZFormat        (XcmsColorFormat)0x00000001
#define XcmsCIEuvYFormat        (XcmsColorFormat)0x00000002
#define XcmsCIExyYFormat        (XcmsColorFormat)0x00000003
#define XcmsCIELabFormat        (XcmsColorFormat)0x00000004
#define XcmsCIELuvFormat        (XcmsColorFormat)0x00000005
#define XcmsTekHVCFormat        (XcmsColorFormat)0x00000006
#define XcmsRGBFormat           (XcmsColorFormat)0x80000000
#define XcmsRGBiFormat          (XcmsColorFormat)0x80000001

#define XcmsInitNone            0x00
#define XcmsInitSuccess         0x01

typedef unsigned int XcmsColorFormat;

typedef double XcmsFloat;

typedef struct {
      unsigned short red;
      unsigned short green;
      unsigned short blue;
} XcmsRGB;
```

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

```
typedef struct {
        XcmsFloat red;
        XcmsFloat green;
        XcmsFloat blue;
} XcmsRGBi;

typedef struct {
        XcmsFloat X;
        XcmsFloat Y;
        XcmsFloat Z;
} XcmsCIEXYZ;

typedef struct {
        XcmsFloat u_prime;
        XcmsFloat v_prime;
        XcmsFloat Y;
} XcmsCIEuvY;

typedef struct {
        XcmsFloat x;
        XcmsFloat y;
        XcmsFloat Y;
} XcmsCIExyY;

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

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

```
typedef struct {
        XcmsFloat L_star;
        XcmsFloat u_star;
        XcmsFloat v_star;
} XcmsCIELuv;

typedef struct {
        XcmsFloat H;
        XcmsFloat V;
        XcmsFloat C;
} XcmsTekHVC;

typedef struct {
        XcmsFloat pad0;
        XcmsFloat pad1;
        XcmsFloat pad2;
        XcmsFloat pad3;
} XcmsPad;
```

6-156

LIBRARIES

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

Page: 247

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

```
typedef struct {
      union {
              XcmsRGB       RGB;
              XcmsRGBi      RGBi;
              XcmsCIEXYZ    CIEXYZ;
              XcmsCIEuvY    CIEuvY;
              XcmsCIExyY    CIExyY;
              XcmsCIELab    CIELab;
              XcmsCIELuv    CIELuv;
              XcmsTekHVC    TekHVC;
              XcmsPad       Pad;
      }     spec;
      unsigned longpixel;
      XcmsColorFormat     format;
} XcmsColor;

typedef struct {
      XcmsColor           screenWhitePt;
      XPointer            functionSet;
      XPointer            screenData;
      unsigned char       state;
      char                pad[3];
} XcmsPerScrnInfo;

typedef void *XcmsCCC;

typedef Status (*XcmsConversionProc)();
typedef XcmsConversionProc *XcmsFuncListPtr;
```

System Data Interfaces 6-157

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

Page: 248

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

```
typedef struct {
        char                    *prefix;
        XcmsColorFormat         id;
        XcmsParseStringProc     parseString;
        XcmsFuncListPtr         to_CIEXYZ;
        XcmsFuncListPtr         from_CIEXYZ;
        int                     inverse_flag;
} XcmsColorSpace;

typedef struct {
        XcmsColorSpace          **DDColorSpaces;
        XcmsScreenInitProc      screenInitProc;
        XcmsScreenFreeProc      screenFreeProc;
} XcmsFunctionSet;
```

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

```
typedef void *XPointer;

#define Bool                   int
#define Status                 int
#define True                   1
#define False                  0
#define QueuedAlready          0
#define QueuedAfterReading     1
#define QueuedAfterFlush       2


#define AllPlanes              ((unsigned long)~0L)
```

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

```
typdef void XExtData;

typdef void XExtCodes;

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

System Data Interfaces 6-159

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

Page: 250

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

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

typedef void *GC;

typedef struct _dummy Visual;
```

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

```
typedef struct _dummy Screen;

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

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

```
typedef struct _dummy ScreenFormat;

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

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

```
typedef struct {
        int family;
        int length;
        char *address;
} XHostAddress;

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

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

Page: 254

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

```
typedef struct {
       int x, y;
       int width, height;
       int border_width;
       Window sibling;
       int stack_mode;
} XWindowChanges;

typedef struct {
       unsigned long pixel;
       unsigned short red, green, blue;
       char flags;
       char pad;
} XColor;

typedef struct {
       short x1, y1, x2, y2;
} XSegment;

typedef struct {
       short x, y;
} XPoint;

typedef struct {
       short x, y;
       unsigned short width, height;
} XRectangle;

typedef struct {
       short x, y;
       unsigned short width, height;
       short angle1, angle2;
} XArc;
```

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

```
typedef struct {
        int key_click_percent;
        int bell_percent;
        int bell_pitch;
        int bell_duration;
        int led;
        int led_mode;
        int key;
        int auto_repeat_mode;
} XKeyboardControl;

typedef struct {
        int key_click_percent;
        int bell_percent;
        unsigned int bell_pitch, bell_duration;
        unsigned long led_mask;
        int global_auto_repeat;
        char auto_repeats[32];
} XKeyboardState;
typedef struct {
        Time time;
        short x, y;
} XTimeCoord;

typedef struct {
        int           max_keypermod;
        KeyCode       *modifiermap;
} XModifierKeymap;

typedef struct _dummy Display;
```

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

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

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

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

Page: 257

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

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

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

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

```
typedef XCrossingEvent XEnterWindowEvent;
typedef XCrossingEvent XLeaveWindowEvent;
typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        int mode;
        int detail;
} XFocusChangeEvent;
typedef XFocusChangeEvent XFocusInEvent;
typedef XFocusChangeEvent XFocusOutEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        char key_vector[32];
} XKeymapEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        int x, y;
        int width, height;
        int count;
} XExposeEvent;
```

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

Page: 259

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

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

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Drawable drawable;
        int major_code;
        int minor_code;
} XNoExposeEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        int state;
} XVisibilityEvent;
```

System Data Interfaces 6-169

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

Page: 260

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

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

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window event;
        Window window;
} XDestroyWindowEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window event;
        Window window;
        Bool from_configure;
} XUnmapEvent;
```

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

```
typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window event;
        Window window;
        Bool override_redirect;
} XMapEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window parent;
        Window window;
} XMapRequestEvent;

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

System Data Interfaces 6-171

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

Page: 262

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

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

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window event;
        Window window;
        int x, y;
} XGravityEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        int width, height;
} XResizeRequestEvent;
```

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

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

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window event;
        Window window;
        int place;
} XCirculateEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window parent;
        Window window;
        int place;
} XCirculateRequestEvent;
```

System Data Interfaces 6-173

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

Page: 264

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

```
typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        Atom atom;
        Time time;
        int state;
} XPropertyEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        Atom selection;
        Time time;
} XSelectionClearEvent;

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

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

Page: 265

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

```
typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window requestor;
        Atom selection;
        Atom target;
        Atom property;
        Time time;
} XSelectionEvent;

typedef struct {
        int type;
        Display *display;
        XID resourceid;
        unsigned long serial;
        unsigned char error_code;
        unsigned char request_code;
        unsigned char minor_code;
} XErrorEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        Atom message_type;
        int format;
        union {
                char b[20];
                short s[10];
                long l[5];
        } data;
} XClientMessageEvent;
```

System Data Interfaces 6-175

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

Page: 266

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

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

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
        int request;
        int first_keycode;
        int count;
} XMappingEvent;

typedef struct {
        int type;
        unsigned long serial;
        Bool send_event;
        Display *display;
        Window window;
} XAnyEvent;
```

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

Page: 267

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

```
typedef union _XEvent {
        int                     type;
        XAnyEvent               xany;
        XKeyEvent               xkey;
        XButtonEvent            xbutton;
        XMotionEvent            xmotion;
        XCrossingEvent          xcrossing;
        XFocusChangeEvent       xfocus;
        XExposeEvent            xexpose;
        XGraphicsExposeEvent    xgraphicsexpose;
        XNoExposeEvent          xnoexpose;
        XVisibilityEvent        xvisibility;
        XCreateWindowEvent      xcreatewindow;
        XDestroyWindowEvent     xdestroywindow;
        XUnmapEvent             xunmap;
        XMapEvent               xmap;
        XMapRequestEvent        xmaprequest;
        XReparentEvent          xreparent;
        XConfigureEvent         xconfigure;
        XGravityEvent           xgravity;
        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;
```

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

Page: 268

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

```
typedef struct {
        short  lbearing;
        short  rbearing;
        short  width;
        short  ascent;
        short  descent;
        unsigned short attributes;
} XCharStruct;

typedef struct {
        Atom name;
        unsigned long card32;
} XFontProp;

typedef struct {
        XExtData      *ext_data;
        Font          fid;
        unsigned      direction;
        unsigned      min_char_or_byte2;
        unsigned      max_char_or_byte2;
        unsigned      min_byte1;
        unsigned      max_byte1;
        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;
```

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

```
typedef struct {
       char *chars;
       int nchars;
       int delta;
       Font font;
} XTextItem;

typedef struct {
       unsigned char 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;
} XEDataObject;

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

typedef struct _dummy XFontSet;
```

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

Page: 270

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

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

typedef struct {
        wchar_t         *chars;
        int             nchars;
        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 XIMPreeditArea          0x0001L
#define XIMPreeditCallbacks     0x0002L
#define XIMPreeditPosition      0x0004L
#define XIMPreeditNothing       0x0008L
#define XIMPreeditNone          0x0010L
#define XIMStatusArea           0x0100L
#define XIMStatusCallbacks      0x0200L
#define XIMStatusNothing        0x0400L
#define XIMStatusNone           0x0800L
```

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

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

System Data Interfaces 6-181

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

Page: 272

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

```
#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;
} XIMCallback;

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;
```

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

```
typedef struct _XIMPreeditDrawCallbackStruct {
        int caret;
        int chg_first;
        int chg_length;
        XIMText *text;
} XIMPreeditDrawCallbackStruct;

typedef enum {
        XIMForwardChar, XIMBackwardChar,
        XIMForwardWord, XIMBackwardWord,
        XIMCaretUp, XIMCaretDown,
        XIMNextLine, XIMPreviousLine,
        XIMLineStart, XIMLineEnd,
        XIMAbsolutePosition,
        XIMDontChange
} XIMCaretDirection;

typedef enum {
        XIMIsInvisible,
        XIMIsPrimary,
        XIMIsSecondary
} XIMCaretStyle;

typedef struct _XIMPreeditCaretCallbackStruct {
        int position;
        XIMCaretDirection direction;
        XIMCaretStyle style;
} XIMPreeditCaretCallbackStruct;
```

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

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

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

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

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

```
typedef int            XrmQuark, *XrmQuarkList;
#define NULLQUARK      ((XrmQuark) 0)

typedef enum {XrmBindTightly, XrmBindLoosely} \
             XrmBinding, *XrmBindingList;

typedef XrmQuark                  XrmName;
typedef XrmQuarkList              XrmNameList;
typedef XrmQuark                  XrmClass;
typedef XrmQuarkList              XrmClassList;
typedef XrmQuark                  XrmRepresentation;

#define XrmStringToName(string)        XrmStringToQuark(string)
#define XrmStringToNameList(str, name)   XrmStringToQuarkList(str, name)
#define XrmStringToClass(class)          XrmStringToQuark(class)
#define XrmStringToClassList(str,class)  XrmStringToQuarkList(str, class)
#define XrmStringToRepresentation(string) XrmStringToQuark(string)

typedef struct {
      unsigned int        size;
      XPointer           addr;
} XrmValue, *XrmValuePtr;

typedef void              *XrmHashBucket;
typedef XrmHashBucket      *XrmHashTable;
typedef XrmHashTable        XrmSearchList[];
typedef void              *XrmDatabase;

#define XrmEnumAllLevels            0
#define XrmEnumOneLevel             1
```

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

```
typedef enum {
        XrmoptionNoArg,
        XrmoptionIsArg,
        XrmoptionStickyArg,
        XrmoptionSepArg,
        XrmoptionResArg,
        XrmoptionSkipArg,
        XrmoptionSkipLine,
        XrmoptionSkipNArgs
} XrmOptionKind;

typedef struct {
        char                *option;
        char                *specifier;
        XrmOptionKind       argKind;
        XPointer            value;
} XrmOptionDescRec, *XrmOptionDescList;
```

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

Page: 277

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

```
#define NoValue        0x0000
#define XValue 0x0001
#define YValue 0x0002
#define WidthValue     0x0004
#define HeightValue    0x0008
#define AllValues      0x000F
#define XNegative      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

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

Page: 278

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

```
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
```

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

```
typedef int XContext;

typedef enum {
        XStringStyle,
        XCompoundTextStyle,
        XTextStyle,
        XStdICCTextStyle
} XICCEncodingStyle;

typedef struct {
        int min_width, min_height;
        int max_width, max_height;
        int width_inc, height_inc;
} XIconSize;

typedef struct {
        char *res_name;
        char *res_class;
} XClassHint;

#define XDestroyImage(ximage)
        ((*((ximage)->f.destroy_image))((ximage)))
#define XGetPixel(ximage, x, y)
        ((*((ximage)->f.get_pixel))((ximage), (x), (y)))
#define XPutPixel(ximage, x, y, pixel)
        ((*((ximage)->f.put_pixel))((ximage), (x), (y), (pixel)))
#define XSubImage(ximage, x, y, width, height)
        ((*((ximage)->f.sub_image))((ximage), (x), (y), (width), (height)))
#define XAddPixel(ximage, value)
        ((*((ximage)->f.add_pixel))((ximage), (value)))

typedef struct _XComposeStatus {
        XPointer compose_ptr;
        int chars_matched;
} XComposeStatus;
```

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

```
#define IsKeypadKey(keysym)
        (((unsigned)(keysym) >= XK_KP_Space) && \
        ((unsigned)(keysym) <= XK_KP_Equal))
#define IsCursorKey(keysym)
        (((unsigned)(keysym) >= XK_Home) && \
        ((unsigned)(keysym) < XK_Select))
#define IsPFKey(keysym)
        (((unsigned)(keysym) >= XK_KP_F1) && \
        ((unsigned)(keysym) <= XK_KP_F4))
#define IsFunctionKey(keysym)
        (((unsigned)(keysym) >= XK_F1) && ((unsigned)(keysym) <= XK_F35))
#define IsMiscFunctionKey(keysym)
        (((unsigned)(keysym) >= XK_Select) && \
        ((unsigned)(keysym) <= XK_Break))
#define IsModifierKey(keysym)
        ((((unsigned)(keysym) >= XK_Shift_L) && \
        ((unsigned)(keysym) <= XK_Hyper_R))
        || ((unsigned)(keysym) == XK_Mode_switch)
        || ((unsigned)(keysym) == XK_Num_Lock))

typedef void *Region;

#define RectangleOut  0
#define RectangleIn           1
#define RectanglePart 2

typedef struct {
        Visual *visual;
        VisualID visualid;
        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**

```
#define VisualNoMask        0x0
#define VisualIDMask        0x1
#define VisualScreenMask            0x2
#define VisualDepthMask             0x4
#define VisualClassMask             0x8
#define VisualRedMaskMask           0x10
#define VisualGreenMaskMask  0x20
#define VisualBlueMaskMask          0x40
#define VisualColormapSizeMask      0x80
#define VisualBitsPerRGBMask 0x100
#define VisualAllMask       0x1FF

typedef struct {
        Colormap            colormap;
        unsigned long       red_max;
        unsigned long       red_mult;
        unsigned long       green_max;
        unsigned long       green_mult;
        unsigned long       blue_max;
        unsigned long       blue_mult;
        unsigned long       base_pixel;
        VisualID            visualid;
        XID                 killid;
} XStandardColormap;

#define ReleaseByFreeingColormap ((XID) 1L)
#define BitmapSuccess       0
#define BitmapOpenFailed            1
#define BitmapFileInvalid           2
#define BitmapNoMemory              3
#define XCSUCCESS                   0
#define XCNOMEM                     1
#define XCNOENT                     2
#define XUniqueContext()    ((XContext) XrmUniqueQuark())
```
M

Page: 282

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>`*****

```
#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>`*****

```
typedef struct _XmDialogShellClassRec      * XmDialogShellWidgetClass;
typedef struct _XmDialogShellRec           * XmDialogShellWidget;
```

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

```
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**

```
#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,
         XmDRAG_MOTION,                 XmDROP_SITE_ENTER,
         XmDROP_SITE_LEAVE,             XmDROP_START,
         XmDROP_FINISH,                 XmDRAG_DROP_FINISH,
         XmOPERATION_CHANGED
         } ;

enum{    XmDROP,                        XmDROP_HELP,
         XmDROP_CANCEL,                 XmDROP_INTERRUPT
         } ;
```

Page: 288

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

```
#define XmDROP_NOOP      0L

enum{   XmBLEND_ALL,                    XmBLEND_STATE_SOURCE,
        XmBLEND_JUST_SOURCE,           XmBLEND_NONE
        } ;

enum{   XmDROP_FAILURE,                 XmDROP_SUCCESS
        } ;

enum{   XmCR_TOP_LEVEL_ENTER,           XmCR_TOP_LEVEL_LEAVE,
        XmCR_DRAG_MOTION,               XmCR_DROP
```

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

```
typedef struct _XmTopLevelLeaveCallbackStruct{
    int                reason;
    XEvent             *event;
    Time               timeStamp;
    Screen             *screen;
    Window             window;
}XmTopLevelLeaveCallbackStruct, *XmTopLevelLeaveCallback;

typedef struct _XmDropSiteEnterCallbackStruct{
    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;
```

System Data Interfaces 6-199

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

Page: 290

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

```
typedef struct _XmOperationChangedCallbackStruct{
    int                reason;
    XEvent             *event;
    Time               timeStamp;
    unsigned char      operation;
    unsigned char      operations;
    unsigned char      dropSiteStatus;
}XmOperationChangedCallbackStruct, *XmOperationChangedCallback;

typedef struct _XmDropStartCallbackStruct{
    int                reason;
    XEvent             *event;
    Time               timeStamp;
    unsigned char      operation;
    unsigned char      operations;
    unsigned char      dropSiteStatus;
    unsigned char      dropAction;
    Position           x, y;
    Window             window;
    Atom               iccHandle;
}XmDropStartCallbackStruct, *XmDropStartCallback;

typedef struct _XmDropFinishCallbackStruct{
    int                reason;
    XEvent             *event;
    Time               timeStamp;
    unsigned char      operation;
    unsigned char      operations;
    unsigned char      dropSiteStatus;
    unsigned char      dropAction;
    unsigned char      completionStatus;
}XmDropFinishCallbackStruct, *XmDropFinishCallback;
typedef struct _XmDragDropFinishCallbackStruct{
    int                reason;
    XEvent             *event;
    Time               timeStamp;
}XmDragDropFinishCallbackStruct, *XmDragDropFinishCallback;
```

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

Page: 291

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

```
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>`*****

```
typedef struct _XmDragOverShellRec            *XmDragOverShellWidget;
typedef struct _XmDragOverShellClassRec       *XmDragOverShellWidgetClass;
```

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

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

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

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

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

```
#define XmCR_DROP_SITE_LEAVE_MESSAGE    1
#define XmCR_DROP_SITE_ENTER_MESSAGE    2
#define XmCR_DROP_SITE_MOTION_MESSAGE   3
#define XmCR_DROP_MESSAGE               4

#define XmNO_DROP_SITE                  1
#define XmINVALID_DROP_SITE             2
#define XmVALID_DROP_SITE               3

enum { XmDRAG_UNDER_NONE, XmDRAG_UNDER_PIXMAP,
    XmDRAG_UNDER_SHADOW_IN, XmDRAG_UNDER_SHADOW_OUT,
    XmDRAG_UNDER_HIGHLIGHT };

enum { XmDROP_SITE_SIMPLE, XmDROP_SITE_COMPOSITE,
    XmDROP_SITE_SIMPLE_CLIP_ONLY = 128,
    XmDROP_SITE_COMPOSITE_CLIP_ONLY };

enum { XmABOVE, XmBELOW };

enum { XmDROP_SITE_ACTIVE, XmDROP_SITE_INACTIVE };

typedef struct _XmDragProcCallbackStruct {
        int                 reason;
        XEvent *            event;
        Time                timeStamp;
        Widget              dragContext;
        Position            x, y;
        unsigned char       dropSiteStatus;
        unsigned char       operation;
        unsigned char       operations;
        Boolean             animate;
} XmDragProcCallbackStruct, * XmDragProcCallback;
```

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

```
typedef struct _XmDropProcCallbackStruct {
        int                 reason;
        XEvent *            event;
        Time                timeStamp;
        Widget              dragContext;
        Position            x, y;
        unsigned char       dropSiteStatus;
        unsigned char       operation;
        unsigned char       operations;
        unsigned char       dropAction;
} XmDropProcCallbackStruct, * XmDropProcCallback;

typedef struct _XmDropSiteVisualsRec {
        Pixel               background;
        Pixel               foreground;
        Pixel               topShadowColor;
        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>`*

```
#define XmTRANSFER_FAILURE 0
#define XmTRANSFER_SUCCESS 1

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


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

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

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

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

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

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;
```

6-206   **LIBRARIES**

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

Page: 297

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

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


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

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

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

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

```
typedef struct _XmMenuShellClassRec       * XmMenuShellWidgetClass;
typedef struct _XmMenuShellWidgetRec      * XmMenuShellWidget;
```

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

```
typedef struct _XmMessageBoxClassRec * XmMessageBoxWidgetClass;
typedef struct _XmMessageBoxRec       * XmMessageBoxWidget;
```

Figure 6-249: `<Mrm/MrmPublic.h>`*, **Part 1 of 3**

```
#define MrmSUCCESS              1
#define MrmCREATE_NEW           3
#define MrmINDEX_RETRY          5
#define MrmINDEX_GT             7
#define MrmINDEX_LT             9
#define MrmPARTIAL_SUCCESS      11

#define MrmFAILURE              0
#define MrmNOT_FOUND            2
#define MrmEXISTS               4
#define MrmNUL_GROUP            6
#define MrmNUL_TYPE             8
#define MrmWRONG_GROUP          10
#define MrmWRONG_TYPE           12
#define MrmOUT_OF_RANGE         14
#define MrmBAD_RECORD           16
#define MrmNULL_DATA            18
#define MrmBAD_DATA_INDEX       2
```

Figure 6-250: `<Mrm/MrmPublic.h>`***, Part 2 of 3**

```
#define MrmNULL_ROUTINE          56
#define MrmVEC_TOO_BIG           58
#define MrmBAD_HIERARCHY         60
#define MrmBAD_CLASS_CODE        62
#define MrmDISPLAY_NOT_OPENED    63
#define MrmEOF                   64
#define MrmUNRESOLVED_REFS       65
#define MrmNcreateCallback       "createCallback"
#define MrmCR_CREATE             XmCR_CREATE


#define MrmwcUnknown             1


#define MrmRtypeMin              1
#define MrmRtypeInteger          1
#define MrmRtypeBoolean          2
#define MrmRtypeChar8            3
#define MrmRtypeChar8Vector      4
#define MrmRtypeCString          5
#define MrmRtypeCStringVector    6
#define MrmRtypeFloat            7
#define MrmRtypeCallback         9
#define MrmRtypePixmapImage      10
#define MrmRtypePixmapDDIF       11
#define MrmRtypeResource         12
#define MrmRtypeNull             13
#define MrmRtypeAddrName         14
#define MrmRtypeIconImage        15
#define MrmRtypeFont             16
#define MrmRtypeFontList         17
#define MrmRtypeColor            18
#define MrmRtypeColorTable       19
#define MrmRtypeAny              20
#define MrmRtypeTransTable       21
#define MrmRtypeClassRecName     22
#define MrmRtypeIntegerVector    23
```

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

Page: 301

Figure 6-251: `<Mrm/MrmPublic.h>`*, **Part 3 of 3**

```
#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
```

Figure 6-252: `<Xm/MwmUtil.h>`*, **Part 1 of 3**

```
typedef struct
{
    long        flags;
    long        functions;
    long        decorations;
    int         input_mode;
    long        status;
} MotifWmHints;

typedef MotifWmHints    MwmHints;

#define MWM_HINTS_FUNCTIONS     (1L << 0)
#define MWM_HINTS_DECORATIONS   (1L << 1)
#define MWM_HINTS_INPUT_MODE    (1L << 2)
#define MWM_HINTS_STATUS        (1L << 3)

#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_RESIZEH       (1L << 2)
#define MWM_DECOR_TITLE         (1L << 3)
#define MWM_DECOR_MENU          (1L << 4)
#define MWM_DECOR_MINIMIZE      (1L << 5)
#define MWM_DECOR_MAXIMIZE      (1L << 6)

#define MWM_INPUT_MODELESS                      0
#define MWM_INPUT_PRIMARY_APPLICATION_MODAL     1
#define MWM_INPUT_SYSTEM_MODAL                  2
#define MWM_INPUT_FULL_APPLICATION_MODAL        3
```

Figure 6-253: `<Xm/MwmUtil.h>`*, **Part 2 of 3**

```
#define MWM_TEAROFF_WINDOW          (1L << 0)
#define MWM_INPUT_APPLICATION_MODAL     MWM_INPUT_PRIMARY_APPLICATION_MODAL

typedef struct
{
    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"
```

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

Page: 304

Figure 6-254: `<Xm/MwmUtil.h>`***, Part 3 of 3**

```
#define _XA_MOTIF_WM_MENU         "_MOTIF_WM_MENU"
#define _XA_MWM_MENU              _XA_MOTIF_WM_MENU

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

typedef PropMotifWmInfo PropMwmInfo;

#define PROP_MOTIF_WM_INFO_ELEMENTS     2
#define PROP_MWM_INFO_ELEMENTS          PROP_MOTIF_WM_INFO_ELEMENTS

#define _XA_MOTIF_WM_INFO         "_MOTIF_WM_INFO"
#define _XA_MWM_INFO              _XA_MOTIF_WM_INFO

#define _XA_MOTIF_BINDINGS        "_MOTIF_BINDINGS"
```

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

```
typedef struct _XmPanedWindowClassRec    *XmPanedWindowWidgetClass;
typedef struct _XmPanedWindowRec         *XmPanedWindowWidget;
```

Figure 6-256: `<Xm/PushB.h>`*

```
typedef struct _XmPushButtonClassRec *XmPushButtonWidgetClass;
typedef struct _XmPushButtonRec       *XmPushButtonWidget;
```

Figure 6-257: `<Xm/PushBG.h>`*

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

Figure 6-258: `<Xm/RepType.h>`*

```
#define XmREP_TYPE_INVALID              0x1FFF

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;
```

6-216   **LIBRARIES**

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

Page: 307

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

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

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

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

Figure 6-265: `<Xm/SeparatoG.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;
```

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**

```
#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
```

Figure 6-274: `<Xm/Xm.h>`**, Part 1 of 14**

```
#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 _XmtStringContextRec *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
        } ;
```

Figure 6-275: `<Xm/Xm.h>`*, **Part 2 of 14**

```
#define XmSTRING_COMPONENT_END          ((XmStringComponentType) 126)
#define XmSTRING_COMPONENT_USER_BEGIN   ((XmStringComponentType) 128)
#define XmSTRING_COMPONENT_USER_END     ((XmStringComponentType) 255)

typedef struct _XmPrimitiveClassRec * XmPrimitiveWidgetClass;
typedef struct _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{   XmPIXELS,                       Xm100TH_MILLIMETERS,
        Xm1000TH_INCHES,                Xm100TH_POINTS,
        Xm100TH_FONT_UNITS
        } ;

enum{   XmDESTROY,                      XmUNMAP,
        XmDO_NOTHING
        } ;

enum{   XmEXPLICIT,                     XmPOINTER
        } ;

enum{   XmNONE,                         XmTAB_GROUP,
        XmSTICKY_TAB_GROUP,             XmEXCLUSIVE_TAB_GROUP
        } ;

#define XmDYNAMIC_DEFAULT_TAB_GROUP     ((XmNavigationType) 255)
```

Figure 6-276: `<Xm/Xm.h>`*, **Part 3 of 14**

```
enum{                        XmBELL = 1
        } ;


enum{    XmNO_ORIENTATION,              XmVERTICAL,
        XmHORIZONTAL
        } ;


enum{    XmWORK_AREA,                   XmMENU_BAR,
        XmMENU_PULLDOWN,               XmMENU_POPUP,
        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
        } ;
```

Page: 314

Figure 6-277: `<Xm/Xm.h>`*, **Part 4 of 14**

```
enum{    XmALIGNMENT_BASELINE_TOP,
         XmALIGNMENT_BASELINE_BOTTOM = 2, XmALIGNMENT_WIDGET_TOP,
         XmALIGNMENT_WIDGET_BOTTOM
         } ;

enum{    XmFRAME_GENERIC_CHILD,             XmFRAME_WORKAREA_CHILD,
         XmFRAME_TITLE_CHILD
         } ;

enum{    XmN_OF_MANY = 1,                   XmONE_OF_MANY
         } ;

enum{    XmATTACH_NONE,                     XmATTACH_FORM,
         XmATTACH_OPPOSITE_FORM,           XmATTACH_WIDGET,
         XmATTACH_OPPOSITE_WIDGET,         XmATTACH_POSITION,
         XmATTACH_SELF
         } ;

enum{    XmRESIZE_NONE,                     XmRESIZE_GROW,
         XmRESIZE_ANY
         } ;
```

Figure 6-278: `<Xm/Xm.h>`*, **Part 5 of 15**

```
enum{    XmCR_NONE,                       XmCR_HELP,
         XmCR_VALUE_CHANGED,              XmCR_INCREMENT,
         XmCR_DECREMENT,                  XmCR_PAGE_INCREMENT,
         XmCR_PAGE_DECREMENT,             XmCR_TO_TOP,
         XmCR_TO_BOTTOM,                  XmCR_DRAG,
         XmCR_ACTIVATE,                   XmCR_ARM,
         XmCR_DISARM,                     XmCR_MAP = 16,
         XmCR_UNMAP,                      XmCR_FOCUS,
         XmCR_LOSING_FOCUS,               XmCR_MODIFYING_TEXT_VALUE,
         XmCR_MOVING_INSERT_CURSOR,       XmCR_EXECUTE,
         XmCR_SINGLE_SELECT,              XmCR_MULTIPLE_SELECT,
         XmCR_EXTENDED_SELECT,            XmCR_BROWSE_SELECT,
         XmCR_DEFAULT_ACTION,             XmCR_CLIPBOARD_DATA_REQUEST,
         XmCR_CLIPBOARD_DATA_DELETE,      XmCR_CASCADING,
         XmCR_OK,                         XmCR_CANCEL,
         XmCR_APPLY = 34,                 XmCR_NO_MATCH,
         XmCR_COMMAND_ENTERED,            XmCR_COMMAND_CHANGED,
         XmCR_EXPOSE,                     XmCR_RESIZE,
         XmCR_INPUT,                      XmCR_GAIN_PRIMARY,
         XmCR_LOSE_PRIMARY,               XmCR_CREATE,
         XmCR_TEAR_OFF_ACTIVATE,          XmCR_TEAR_OFF_DEACTIVATE,
         XmCR_OBSCURED_TRAVERSAL
         } ;

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

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

System Data Interfaces **6-225**

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

Page: 316

Figure 6-279: <Xm/Xm.h>*, **Part 6 of 14**

```
typedef struct
{
    int     reason;
    XEvent  *event;
    Window  window;
} XmDrawingAreaCallbackStruct;

typedef struct
{
    int     reason;
    XEvent  *event;
    Window  window;
    int     click_count;
} XmDrawnButtonCallbackStruct;

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

Figure 6-280: <Xm/Xm.h>*, **Part 7 of 14**

```
typedef struct
{
    int reason;
    XEvent * event;
    int value;
    int pixel;
} XmScrollBarCallbackStruct;

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

typedef struct
{
    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
```

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

```
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;

enum{   XmMULTICLICK_DISCARD,          XmMULTICLICK_KEEP
        } ;

enum{   XmSHADOW_IN = 7,               XmSHADOW_OUT
        } ;
```

Figure 6-282: `<Xm/Xm.h>`***, Part 9 of 14**

```
enum{    XmARROW_UP,                      XmARROW_DOWN,
         XmARROW_LEFT,                    XmARROW_RIGHT
         } ;

enum{    XmNO_LINE,                       XmSINGLE_LINE,
         XmDOUBLE_LINE,                   XmSINGLE_DASHED_LINE,
         XmDOUBLE_DASHED_LINE,            XmSHADOW_ETCHED_IN,
         XmSHADOW_ETCHED_OUT,             XmSHADOW_ETCHED_IN_DASH,
         XmSHADOW_ETCHED_OUT_DASH,        XmINVALID_SEPARATOR_TYPE
         } ;

enum{    XmPIXMAP = 1,                    XmSTRING
         } ;

enum{    XmWINDOW,
         XmCURSOR = 2
         } ;

enum{    XmMAX_ON_TOP,                    XmMAX_ON_BOTTOM,
         XmMAX_ON_LEFT,                   XmMAX_ON_RIGHT
         } ;

enum{    XmSINGLE_SELECT,                 XmMULTIPLE_SELECT,
         XmEXTENDED_SELECT,              XmBROWSE_SELECT
         } ;

enum{    XmSTATIC,                        XmDYNAMIC
         } ;

enum{    XmVARIABLE,                      XmCONSTANT,
         XmRESIZE_IF_POSSIBLE
         } ;

enum{    XmAUTOMATIC,                     XmAPPLICATION_DEFINED
         } ;
```

System Data Interfaces 6-229

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

Page: 320

Figure 6-283: `<Xm/Xm.h>`*, **Part 10 of 14**

```
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
        } ;

enum{   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
```

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

Page: 321

Figure 6-284: `<Xm/Xm.h>`*, **Part 11 of 14**

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

typedef enum{
        XmHIGHLIGHT_NORMAL,                 XmHIGHLIGHT_SELECTED,
        XmHIGHLIGHT_SECONDARY_SELECTED
        } XmHighlightMode ;

typedef struct {
    char *ptr;
    int length;
    XmTextFormat format;
} XmTextBlockRec, *XmTextBlock;

typedef struct
{
    int reason;
    XEvent  *event;
    Boolean doit;
    long currInsert, newInsert;
    long startPos, endPos;
    XmTextBlock text;
} XmTextVerifyCallbackStruct, *XmTextVerifyPtr;

typedef struct {
    wchar_t *wcsptr;
    int length;
} XmTextBlockRecWcs, *XmTextBlockWcs;
```

System Data Interfaces 6-231

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

Page: 322

Figure 6-285: `<Xm/Xm.h>`**, Part 12 of 14**

```
typedef struct
{
    int reason;
    XEvent  *event;
    Boolean doit;
    long currInsert, newInsert;
    long startPos, endPos;
    XmTextBlockWcs text;
} XmTextVerifyCallbackStructWcs, *XmTextVerifyPtrWcs;

#define XmTextGetTopPosition                XmTextGetTopCharacter
#define XmTextSetTopPosition                XmTextSetTopCharacter

#define XmCOPY_FAILED          0
#define XmCOPY_SUCCEEDED       1
#define XmCOPY_TRUNCATED       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_TEXT,
        XmDIALOG_SEPARATOR,        XmDIALOG_DIR_LIST,
        XmDIALOG_DIR_LIST_LABEL
        } ;

#define XmDIALOG_HISTORY_LIST        XmDIALOG_LIST
#define XmDIALOG_PROMPT_LABEL        XmDIALOG_SELECTION_LABEL
#define XmDIALOG_VALUE_TEXT          XmDIALOG_TEXT
#define XmDIALOG_COMMAND_TEXT        XmDIALOG_TEXT
#define XmDIALOG_FILE_LIST           XmDIALOG_LIST
#define XmDIALOG_FILE_LIST_LABEL     XmDIALOG_LIST_LABEL
```

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

Page: 323

Figure 6-286: `<Xm/Xm.h>`*, **Part 13 of 14**

```
enum{    XmDIALOG_MODELESS,                 XmDIALOG_PRIMARY_APPLICATION_MODAL,
         XmDIALOG_FULL_APPLICATION_MODAL,XmDIALOG_SYSTEM_MODAL
         } ;

#define XmDIALOG_APPLICATION_MODAL      XmDIALOG_PRIMARY_APPLICATION_MODAL

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{    XmDIALOG_WORK_AREA,           XmDIALOG_PROMPT,
         XmDIALOG_SELECTION,           XmDIALOG_COMMAND,
         XmDIALOG_FILE_SELECTION
         } ;

enum{    XmDIALOG_TEMPLATE,            XmDIALOG_ERROR,
         XmDIALOG_INFORMATION,         XmDIALOG_MESSAGE,
         XmDIALOG_QUESTION,            XmDIALOG_WARNING,
         XmDIALOG_WORKING
         } ;

typedef enum{
         XmVISIBILITY_UNOBSCURED,      XmVISIBILITY_PARTIALLY_OBSCURED,
         XmVISIBILITY_FULLY_OBSCURED
         } XmVisibility ;

typedef enum{
         XmTRAVERSE_CURRENT,           XmTRAVERSE_NEXT,
         XmTRAVERSE_PREV,              XmTRAVERSE_HOME,
         XmTRAVERSE_NEXT_TAB_GROUP,    XmTRAVERSE_PREV_TAB_GROUP,
         XmTRAVERSE_UP,                XmTRAVERSE_DOWN,
         XmTRAVERSE_LEFT,              XmTRAVERSE_RIGHT
         } XmTraversalDirection ;
```

Figure 6-287: `<Xm/Xm.h>`*, **Part 14 of 14**

```
typedef struct _XmTraverseObscuredCallbackStruct
{       int                     reason ;
        XEvent *                event ;
        Widget                  traversal_destination ;
        XmTraversalDirection    direction ;
        } XmTraverseObscuredCallbackStruct ;

typedef unsigned char   XmNavigationType;
typedef unsigned char XmButtonType;
typedef XmButtonType * XmButtonTypeTable;
typedef KeySym * XmKeySymTable;
typedef XmStringCharSet * XmStringCharSetTable;

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**

```
#define XmS ""
#define XmCAccelerator "Accelerator"
#define XmCAcceleratorText "AcceleratorText"
#define XmCAdjustLast "AdjustLast"
#define XmCAdjustMargin "AdjustMargin"
#define XmCAlignment "Alignment"
#define XmCAllowOverlap "AllowOverlap"
#define XmCAnimationMask "AnimationMask"
#define XmCAnimationPixmap "AnimationPixmap"
#define XmCAnimationPixmapDepth "AnimationPixmapDepth"
#define XmCAnimationStyle "AnimationStyle"
#define XmCApplyLabelString "ApplyLabelString"
#define XmCArmCallback "ArmCallback"
#define XmCArmColor "ArmColor"
#define XmCArmPixmap "ArmPixmap"
#define XmCArrowDirection "ArrowDirection"
#define XmCAttachment "Attachment"
#define XmCAudibleWarning "AudibleWarning"
#define XmCAutoShowCursorPosition "AutoShowCursorPosition"
#define XmCAutoUnmanage "AutoUnmanage"
#define XmCAutomaticSelection "AutomaticSelection"
#define XmCAvailability "Availability"
#define XmCBackgroundPixmap "BackgroundPixmap"
#define XmCBlendModel "BlendModel"
#define XmCBlinkRate "BlinkRate"
#define XmCBottomShadowColor "BottomShadowColor"
#define XmCBottomShadowPixmap "BottomShadowPixmap"
#define XmCButtonAcceleratorText "ButtonAcceleratorText"
#define XmCButtonAccelerators "ButtonAccelerators"
#define XmCButtonCount "ButtonCount"
#define XmCButtonFontList "ButtonFontList"
#define XmCButtonMnemonicCharSets "ButtonMnemonicCharSets"
#define XmCButtonMnemonics "ButtonMnemonics"
#define XmCButtonSet "ButtonSet"
#define XmCButtonType "ButtonType"
```

Figure 6-289: `<Xm/XmStrDefs.h>`*, Part 2 of 34*

```
#define XmCButtons "Buttons"
#define XmCCancelLabelString "CancelLabelString"
#define XmCChildHorizontalAlignment "ChildHorizontalAlignment"
#define XmCChildHorizontalSpacing "ChildHorizontalSpacing"
#define XmCChildPlacement "ChildPlacement"
#define XmCChildType "ChildType"
#define XmCChildVerticalAlignment "ChildVerticalAlignment"
#define XmCChildren "Children"
#define XmCClientData "ClientData"
#define XmCClipWindow "ClipWindow"
#define XmCColumns "Columns"
#define XmCCommandWindow "CommandWindow"
#define XmCCommandWindowLocation "CommandWindowLocation"
#define XmCConvertProc "ConvertProc"
#define XmCCursorBackground "CursorBackground"
#define XmCCursorForeground "CursorForeground"
#define XmCCursorPosition "CursorPosition"
#define XmCCursorPositionVisible "CursorPositionVisible"
#define XmCDarkThreshold "DarkThreshold"
#define XmCDecimalPoints "DecimalPoints"
#define XmCDefaultButtonShadowThickness "DefaultButtonShadowThickness"
#define XmCDefaultButtonType "DefaultButtonType"
#define XmCDefaultCopyCursorIcon "DefaultCopyCursorIcon"
#define XmCDefaultFontList "DefaultFontList"
#define XmCDefaultInvalidCursorIcon "DefaultInvalidCursorIcon"
#define XmCDefaultLinkCursorIcon "DefaultLinkCursorIcon"
#define XmCDefaultMoveCursorIcon "DefaultMoveCursorIcon"
#define XmCDefaultNoneCursorIcon "DefaultNoneCursorIcon"
#define XmCDefaultPosition "DefaultPosition"
#define XmCDefaultSourceCursorIcon "DefaultSourceCursorIcon"
#define XmCDefaultValidCursorIcon "DefaultValidCursorIcon"
#define XmCDeleteResponse "DeleteResponse"
#define XmCDesktopParent "DesktopParent"
#define XmCDialogStyle "DialogStyle"
#define XmCDialogTitle "DialogTitle"
```

Figure 6-290: `<Xm/XmStrDefs.h>`*, **Part 3 of 34**

```
#define XmCDialogType "DialogType"
#define XmCDirListItemCount "DirListItemCount"
#define XmCDirListItems "DirListItems"
#define XmCDirListLabelString "DirListLabelString"
#define XmCDirMask "DirMask"
#define XmCDirSearchProc "DirSearchProc"
#define XmCDirSpec "DirSpec"
#define XmCDirectory "Directory"
#define XmCDirectoryValid "DirectoryValid"
#define XmCDisarmCallback "DisarmCallback"
#define XmCDoubleClickInterval "DoubleClickInterval"
#define XmCDragContextClass "DragContextClass"
#define XmCDragDropFinishCallback "DragDropFinishCallback"
#define XmCDragIconClass "DragIconClass"
#define XmCDragInitiatorProtocolStyle "DragInitiatorProtocolStyle"
#define XmCDragMotionCallback "DragMotionCallback"
#define XmCDragOperations "DragOperations"
#define XmCDragOverMode "DragOverMode"
#define XmCDragProc "DragProc"
#define XmCDragReceiverProtocolStyle "DragReceiverProtocolStyle"
#define XmCDropProc "DropProc"
#define XmCDropRectangles "DropRectangles"
#define XmCDropSiteActivity "DropSiteActivity"
#define XmCDropSiteEnterCallback "DropSiteEnterCallback"
#define XmCDropSiteLeaveCallback "DropSiteLeaveCallback"
#define XmCDropSiteManagerClass "DropSiteManagerClass"
#define XmCDropSiteOperations "DropSiteOperations"
#define XmCDropSiteType "DropSiteType"
#define XmCDropStartCallback "DropStartCallback"
#define XmCDropTransferClass "DropTransferClass"
#define XmCDropTransfers "DropTransfers"
#define XmCEditable "Editable"
#define XmCEntryBorder "EntryBorder"
#define XmCEntryClass "EntryClass"
#define XmCExportTargets "ExportTargets"
```

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

Page: 328

Figure 6-291: `<Xm/XmStrDefs.h>`*, Part 4 of 34*

```
#define XmCExposeCallback "ExposeCallback"
#define XmCExtensionType "ExtensionType"
#define XmCFileListItemCount "FileListItemCount"
#define XmCFileListItems "FileListItems"
#define XmCFileListLabelString "FileListLabelString"
#define XmCFileSearchProc "FileSearchProc"
#define XmCFileTypeMask "FileTypeMask"
#define XmCFillOnArm "FillOnArm"
#define XmCFillOnSelect "FillOnSelect"
#define XmCFilterLabelString "FilterLabelString"
#define XmCFontList "FontList"
#define XmCForegroundThreshold "ForegroundThreshold"
#define XmCHelpLabelString "HelpLabelString"
#define XmCHighlightColor "HighlightColor"
#define XmCHighlightOnEnter "HighlightOnEnter"
#define XmCHighlightPixmap "HighlightPixmap"
#define XmCHighlightThickness "HighlightThickness"
#define XmCHorizontalFontUnit "HorizontalFontUnit"
#define XmCHorizontalScrollBar "HorizontalScrollBar"
#define XmCHot "Hot"
#define XmCICCHandle "ICCHandle"
#define XmCImportTargets "ImportTargets"
#define XmCIncrement "Increment"
#define XmCIncremental "Incremental"
#define XmCIndicatorOn "IndicatorOn"
#define XmCIndicatorSize "IndicatorSize"
#define XmCIndicatorType "IndicatorType"
#define XmCInitialDelay "InitialDelay"
#define XmCInitialFocus "InitialFocus"
#define XmCInputCreate "InputCreate"
#define XmCInputMethod "InputMethod"
#define XmCInvalidCursorForeground "InvalidCursorForeground"
#define XmCIsAligned "IsAligned"
#define XmCIsHomogeneous "IsHomogeneous"
#define XmCItemCount "ItemCount"
```

Figure 6-292: `<Xm/XmStrDefs.h>`*, **Part 5 of 34**

```
#define XmCItems "Items"
#define XmCKeyboardFocusPolicy "KeyboardFocusPolicy"
#define XmCLabelFontList "LabelFontList"
#define XmCLabelInsensitivePixmap "LabelInsensitivePixmap"
#define XmCLabelPixmap "LabelPixmap"
#define XmCLabelString "LabelString"
#define XmCLabelType "LabelType"
#define XmCLightThreshold "LightThreshold"
#define XmCListLabelString "ListLabelString"
#define XmCListMarginHeight "ListMarginHeight"
#define XmCListMarginWidth "ListMarginWidth"
#define XmCListSizePolicy "ListSizePolicy"
#define XmCListSpacing "ListSpacing"
#define XmCListUpdated "ListUpdated"
#define XmCLogicalParent "LogicalParent"
#define XmCMainWindowMarginHeight "MainWindowMarginHeight"
#define XmCMainWindowMarginWidth "MainWindowMarginWidth"
#define XmCMappingDelay "MappingDelay"
#define XmCMarginBottom "MarginBottom"
#define XmCMarginHeight "MarginHeight"
#define XmCMarginLeft "MarginLeft"
#define XmCMarginRight "MarginRight"
#define XmCMarginTop "MarginTop"
#define XmCMarginWidth "MarginWidth"
#define XmCMask "Mask"
#define XmCMaxItems "MaxItems"
#define XmCMaxLength "MaxLength"
#define XmCMaxValue "MaxValue"
#define XmCMaximum "Maximum"
#define XmCMenuBar "MenuBar"
#define XmCMenuPost "MenuPost"
#define XmCMenuWidget "MenuWidget"
#define XmCMessageProc "MessageProc"
#define XmCMessageWindow "MessageWindow"
#define XmCMinimizeButtons "MinimizeButtons"
```

Figure 6-293: `<Xm/XmStrDefs.h>`***, Part 6 of 34**

```
#define XmCMinimum "Minimum"
#define XmCMnemonic "Mnemonic"
#define XmCMnemonicCharSet "MnemonicCharSet"
#define XmCMoveOpaque "MoveOpaque"
#define XmCMultiClick "MultiClick"
#define XmCMustMatch "MustMatch"
#define XmCMwmDecorations "MwmDecorations"
#define XmCMwmFunctions "MwmFunctions"
#define XmCMwmInputMode "MwmInputMode"
#define XmCMwmMenu "MwmMenu"
#define XmCMwmMessages "MwmMessages"
#define XmCNavigationType "NavigationType"
#define XmCNeedsMotion "NeedsMotion"
#define XmCNoMatchString "NoMatchString"
#define XmCNoResize "NoResize"
#define XmCNoneCursorForeground "NoneCursorForeground"
#define XmCNotifyProc "NotifyProc"
#define XmCNumChildren "NumChildren"
#define XmCNumColumns "NumColumns"
#define XmCNumDropRectangles "NumDropRectangles"
#define XmCNumDropTransfers "NumDropTransfers"
#define XmCNumExportTargets "NumExportTargets"
#define XmCNumImportTargets "NumImportTargets"
#define XmCOffset "Offset"
#define XmCOkLabelString "OkLabelString"
#define XmCOperationChangedCallback "OperationChangedCallback"
#define XmCOperationCursorIcon "OperationCursorIcon"
#define XmCOptionLabel "OptionLabel"
#define XmCOptionMnemonic "OptionMnemonic"
#define XmCOutputCreate "OutputCreate"
#define XmCPacking "Packing"
#define XmCPageIncrement "PageIncrement"
#define XmCPaneMaximum "PaneMaximum"
#define XmCPaneMinimum "PaneMinimum"
#define XmCPattern "Pattern"
```

6-240 **LIBRARIES**

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

Page: 331

Figure 6-294: `<Xm/XmStrDefs.h>`*, **Part 7 of 34**

```
#define XmCPendingDelete "PendingDelete"
#define XmCPopupEnabled "PopupEnabled"
#define XmCPositionIndex "PositionIndex"
#define XmCPostFromButton "PostFromButton"
#define XmCPostFromCount "PostFromCount"
#define XmCPostFromList "PostFromList"
#define XmCPreeditType "PreeditType"
#define XmCProcessingDirection "ProcessingDirection"
#define XmCPromptString "PromptString"
#define XmCProtocolCallback "ProtocolCallback"
#define XmCPushButtonEnabled "PushButtonEnabled"
#define XmCQualifySearchDataProc "QualifySearchDataProc"
#define XmCRadioAlwaysOne "RadioAlwaysOne"
#define XmCRadioBehavior "RadioBehavior"
#define XmCRecomputeSize "RecomputeSize"
#define XmCRectangles "Rectangles"
#define XmCRepeatDelay "RepeatDelay"
#define XmCResizeCallback "ResizeCallback"
#define XmCResizeHeight "ResizeHeight"
#define XmCResizePolicy "ResizePolicy"
#define XmCResizeWidth "ResizeWidth"
#define XmCRowColumnType "RowColumnType"
#define XmCRows "Rows"
#define XmCRubberPositioning "RubberPositioning"
#define XmCSashHeight "SashHeight"
#define XmCSashIndent "SashIndent"
#define XmCSashWidth "SashWidth"
#define XmCScaleHeight "ScaleHeight"
#define XmCScaleMultiple "ScaleMultiple"
#define XmCScaleWidth "ScaleWidth"
#define XmCScroll "Scroll"
#define XmCScrollBarDisplayPolicy "ScrollBarDisplayPolicy"
#define XmCScrollBarPlacement "ScrollBarPlacement"
#define XmCScrollSide "ScrollSide"
#define XmCScrolledWindowMarginHeight "ScrolledWindowMarginHeight"
```

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

Page: 332

Figure 6-295: `<Xm/XmStrDefs.h>`***, Part 8 of 34**

```
#define XmCScrolledWindowMarginWidth "ScrolledWindowMarginWidth"
#define XmCScrollingPolicy "ScrollingPolicy"
#define XmCSelectColor "SelectColor"
#define XmCSelectInsensitivePixmap "SelectInsensitivePixmap"
#define XmCSelectPixmap "SelectPixmap"
#define XmCSelectThreshold "SelectThreshold"
#define XmCSelectedItemCount "SelectedItemCount"
#define XmCSelectedItems "SelectedItems"
#define XmCSelectionArrayCount "SelectionArrayCount"
#define XmCSelectionLabelString "SelectionLabelString"
#define XmCSelectionPolicy "SelectionPolicy"
#define XmCSeparatorOn "SeparatorOn"
#define XmCSeparatorType "SeparatorType"
#define XmCSet "Set"
#define XmCShadowThickness "ShadowThickness"
#define XmCShadowType "ShadowType"
#define XmCShellUnitType "ShellUnitType"
#define XmCShowArrows "ShowArrows"
#define XmCShowAsDefault "ShowAsDefault"
#define XmCShowSeparator "ShowSeparator"
#define XmCShowValue "ShowValue"
#define XmCSimpleCheckBox "SimpleCheckBox"
#define XmCSimpleMenuBar "SimpleMenuBar"
#define XmCSimpleOptionMenu "SimpleOptionMenu"
#define XmCSimplePopupMenu "SimplePopupMenu"
#define XmCSimplePulldownMenu "SimplePulldownMenu"
#define XmCSimpleRadioBox "SimpleRadioBox"
#define XmCSizePolicy "SizePolicy"
#define XmCSliderSize "SliderSize"
#define XmCSource "Source"
#define XmCSourceCursorIcon "SourceCursorIcon"
#define XmCSourceIsExternal "SourceIsExternal"
#define XmCSourcePixmapIcon "SourcePixmapIcon"
#define XmCSourceWidget "SourceWidget"
#define XmCSourceWindow "SourceWindow"
```

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

Page: 333

Figure 6-296: `<Xm/XmStrDefs.h>`*, **Part 9 of 34**

```
#define XmCSpacing "Spacing"
#define XmCStartTime "StartTime"
#define XmCStateCursorIcon "StateCursorIcon"
#define XmCStringDirection "StringDirection"
#define XmCTearOffModel "TearOffModel"
#define XmCTextFontList "TextFontList"
#define XmCTextString "TextString"
#define XmCTextValue "TextValue"
#define XmCTitleString "TitleString"
#define XmCTopCharacter "TopCharacter"
#define XmCTopItemPosition "TopItemPosition"
#define XmCTopLevelEnterCallback "TopLevelEnterCallback"
#define XmCTopLevelLeaveCallback "TopLevelLeaveCallback"
#define XmCTopShadowColor "TopShadowColor"
#define XmCTopShadowPixmap "TopShadowPixmap"
#define XmCTransferProc "TransferProc"
#define XmCTransferStatus "TransferStatus"
#define XmCTraversalOn "TraversalOn"
#define XmCTraversalType "TraversalType"
#define XmCTreeUpdateProc "TreeUpdateProc"
#define XmCTroughColor "TroughColor"
#define XmCUnitType "UnitType"
#define XmCUnpostBehavior "UnpostBehavior"
#define XmCUnselectPixmap "UnselectPixmap"
#define XmCUpdateSliderSize "UpdateSliderSize"
#define XmCUseAsyncGeometry "UseAsyncGeometry"
#define XmCUserData "UserData"
#define XmCValidCursorForeground "ValidCursorForeground"
#define XmCValueChangedCallback "ValueChangedCallback"
#define XmCValueWcs "ValueWcs"
#define XmCVerifyBell "VerifyBell"
#define XmCVerticalAlignment "VerticalAlignment"
#define XmCVerticalFontUnit "VerticalFontUnit"
#define XmCVerticalScrollBar "VerticalScrollBar"
```

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

Page: 334

Figure 6-297: `<Xm/XmStrDefs.h>`***, Part 10 of 34**

```
#define XmCVisibleItemCount "VisibleItemCount"
#define XmCVisibleWhenOff "VisibleWhenOff"
#define XmCVisualPolicy "VisualPolicy"
#define XmCWhichButton "WhichButton"
#define XmCWordWrap "WordWrap"
#define XmCWorkWindow "WorkWindow"
#define XmCXmString "XmString"
#define XmNaccelerator "accelerator"
#define XmNacceleratorText "acceleratorText"
#define XmNactivateCallback "activateCallback"
#define XmNadjustLast "adjustLast"
#define XmNadjustMargin "adjustMargin"
#define XmNalignment "alignment"
#define XmNallowOverlap "allowOverlap"
#define XmNallowResize "allowResize"
#define XmNanimationMask "animationMask"
#define XmNanimationPixmap "animationPixmap"
#define XmNanimationPixmapDepth "animationPixmapDepth"
#define XmNanimationStyle "animationStyle"
#define XmNapplyCallback "applyCallback"
#define XmNapplyLabelString "applyLabelString"
#define XmNarmCallback "armCallback"
#define XmNarmColor "armColor"
#define XmNarmPixmap "armPixmap"
#define XmNarrowDirection "arrowDirection"
#define XmNattachment "attachment"
#define XmNaudibleWarning "audibleWarning"
#define XmNautoShowCursorPosition "autoShowCursorPosition"
#define XmNautoUnmanage "autoUnmanage"
#define XmNautomaticSelection "automaticSelection"
#define XmNavailability "availability"
#define XmNblendModel "blendModel"
#define XmNblinkRate "blinkRate"
#define XmNbottomAttachment "bottomAttachment"
#define XmNbottomOffset "bottomOffset"
```

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

Page: 335

Figure 6-298: `<Xm/XmStrDefs.h>`*, **Part 11 of 34**

```
#define XmNbottomPosition "bottomPosition"
#define XmNbottomShadowColor "bottomShadowColor"
#define XmNbottomShadowPixmap "bottomShadowPixmap"
#define XmNbottomWidget "bottomWidget"
#define XmNbrowseSelectionCallback "browseSelectionCallback"
#define XmNbuttonAcceleratorText "buttonAcceleratorText"
#define XmNbuttonAccelerators "buttonAccelerators"
#define XmNbuttonCount "buttonCount"
#define XmNbuttonFontList "buttonFontList"
#define XmNbuttonMnemonicCharSets "buttonMnemonicCharSets"
#define XmNbuttonMnemonics "buttonMnemonics"
#define XmNbuttonSet "buttonSet"
#define XmNbuttonType "buttonType"
#define XmNbuttons "buttons"
#define XmNcancelButton "cancelButton"
#define XmNcancelCallback "cancelCallback"
#define XmNcancelLabelString "cancelLabelString"
#define XmNcascadePixmap "cascadePixmap"
#define XmNcascadingCallback "cascadingCallback"
#define XmNchildHorizontalAlignment "childHorizontalAlignment"
#define XmNchildHorizontalSpacing "childHorizontalSpacing"
#define XmNchildPlacement "childPlacement"
#define XmNchildPosition "childPosition"
#define XmNchildType "childType"
#define XmNchildVerticalAlignment "childVerticalAlignment"
#define XmNclientData "clientData"
#define XmNclipWindow "clipWindow"
#define XmNcolumns "columns"
#define XmNcommand "command"
#define XmNcommandChangedCallback "commandChangedCallback"
#define XmNcommandEnteredCallback "commandEnteredCallback"
#define XmNcommandWindow "commandWindow"
#define XmNcommandWindowLocation "commandWindowLocation"
#define XmNconvertProc "convertProc"
#define XmNcursorBackground "cursorBackground"
```

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

Page: 336

Figure 6-299: `<Xm/XmStrDefs.h>`***, Part 12 of 34**

```
#define XmNcursorForeground "cursorForeground"
#define XmNcursorPosition "cursorPosition"
#define XmNcursorPositionVisible "cursorPositionVisible"
#define XmNdarkThreshold "darkThreshold"
#define XmNdecimalPoints "decimalPoints"
#define XmNdecrementCallback "decrementCallback"
#define XmNdefaultActionCallback "defaultActionCallback"
#define XmNdefaultButton "defaultButton"
#define XmNdefaultButtonShadowThickness "defaultButtonShadowThickness"
#define XmNdefaultButtonType "defaultButtonType"
#define XmNdefaultCopyCursorIcon "defaultCopyCursorIcon"
#define XmNdefaultFontList "defaultFontList"
#define XmNdefaultInvalidCursorIcon "defaultInvalidCursorIcon"
#define XmNdefaultLinkCursorIcon "defaultLinkCursorIcon"
#define XmNdefaultMoveCursorIcon "defaultMoveCursorIcon"
#define XmNdefaultNoneCursorIcon "defaultNoneCursorIcon"
#define XmNdefaultPosition "defaultPosition"
#define XmNdefaultSourceCursorIcon "defaultSourceCursorIcon"
#define XmNdefaultValidCursorIcon "defaultValidCursorIcon"
#define XmNdeleteResponse "deleteResponse"
#define XmNdesktopParent "desktopParent"
#define XmNdialogStyle "dialogStyle"
#define XmNdialogTitle "dialogTitl 1.65  1.65
```

Figure 6-300: `<Xm/XmStrDefs.h>`*, **Part 13 of 34**

```
#define XmNdragContextClass "dragContextClass"
#define XmNdragDropFinishCallback "dragDropFinishCallback"
#define XmNdragIconClass "dragIconClass"
#define XmNdragInitiatorProtocolStyle "dragInitiatorProtocolStyle"
#define XmNdragMotionCallback "dragMotionCallback"
#define XmNdragOperations "dragOperations"
#define XmNdragOverMode "dragOverMode"
#define XmNdragProc "dragProc"
#define XmNdragReceiverProtocolStyle "dragReceiverProtocolStyle"
#define XmNdropFinishCallback "dropFinishCallback"
#define XmNdropProc "dropProc"
#define XmNdropRectangles "dropRectangles"
#define XmNdropSiteActivity "dropSiteActivity"
#define XmNdropSiteEnterCallback "dropSiteEnterCallback"
#define XmNdropSiteLeaveCallback "dropSiteLeaveCallback"
#define XmNdropSiteManagerClass "dropSiteManagerClass"
#define XmNdropSiteOperations "dropSiteOperations"
#define XmNdropSiteType "dropSiteType"
#define XmNdropStartCallback "dropStartCallback"
#define XmNdropTransferClass "dropTransferClass"
#define XmNdropTransfers "dropTransfers"
#define XmNeditMode "editMode"
#define XmNeditable "editable"
#define XmNentryAlignment "entryAlignment"
#define XmNentryBorder "entryBorder"
#define XmNentryCallback "entryCallback"
#define XmNentryClass "entryClass"
#define XmNentryVerticalAlignment "entryVerticalAlignment"
#define XmNexportTargets "exportTargets"
#define XmNexposeCallback "exposeCallback"
#define XmNextendedSelectionCallback "extendedSelectionCallback"
#define XmNextensionType "extensionType"
#define XmNfileListItemCount "fileListItemCount"
#define XmNfileListItems "fileListItems"
#define XmNfileListLabelString "fileListLabelString"
```

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

Page: 338

Figure 6-301: `<Xm/XmStrDefs.h>`*, Part 14 of 34*

```
#define XmNfileSearchProc "fileSearchProc"
#define XmNfileTypeMask "fileTypeMask"
#define XmNfillOnArm "fillOnArm"
#define XmNfillOnSelect "fillOnSelect"
#define XmNfilterLabelString "filterLabelString"
#define XmNfocusCallback "focusCallback"
#define XmNfocusMovedCallback "focusMovedCallback"
#define XmNfocusPolicyChanged "focusPolicyChanged"
#define XmNfontList "fontList"
#define XmNforegroundThreshold "foregroundThreshold"
#define XmNfractionBase "fractionBase"
#define XmNgainPrimaryCallback "gainPrimaryCallback"
#define XmNhelpCallback "helpCallback"
#define XmNhelpLabelString "helpLabelString"
#define XmNhighlightColor "highlightColor"
#define XmNhighlightOnEnter "highlightOnEnter"
#define XmNhighlightPixmap "highlightPixmap"
#define XmNhighlightThickness "highlightThickness"
#define XmNhistoryItemCount "historyItemCount"
#define XmNhistoryItems "historyItems"
#define XmNhistoryMaxItems "historyMaxItems"
#define XmNhistoryVisibleItemCount "historyVisibleItemCount"
#define XmNhorizontalFontUnit "horizontalFontUnit"
#define XmNhorizontalScrollBar "horizontalScrollBar"
#define XmNhorizontalSpacing "horizontalSpacing"
#define XmNhotX "hotX"
#define XmNhotY "hotY"
#define XmNiccHandle "iccHandle"
#define XmNimportTargets "importTargets"
#define XmNincrement "increment"
#define XmNincrementCallback "incrementCallback"
#define XmNincremental "incremental"
#define XmNindicatorOn "indicatorOn"
#define XmNindicatorSize "indicatorSize"
#define XmNindicatorType "indicatorType"
```

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

Page: 339

Figure 6-302: `<Xm/XmStrDefs.h>`*, **Part 15 of 34**

```
#define XmNinitialDelay "initialDelay"
#define XmNinitialFocus "initialFocus"
#define XmNinputCallback "inputCallback"
#define XmNinputCreate "inputCreate"
#define XmNinputMethod "inputMethod"
#define XmNinvalidCursorForeground "invalidCursorForeground"
#define XmNisAligned "isAligned"
#define XmNisHomogeneous "isHomogeneous"
#define XmNitemCount "itemCount"
#define XmNitems "items"
#define XmNkeyboardFocusPolicy "keyboardFocusPolicy"
#define XmNlabelFontList "labelFontList"
#define XmNlabelInsensitivePixmap "labelInsensitivePixmap"
#define XmNlabelPixmap "labelPixmap"
#define XmNlabelString "labelString"
#define XmNlabelType "labelType"
#define XmNleftAttachment "leftAttachment"
#define XmNleftOffset "leftOffset"
#define XmNleftPosition "leftPosition"
#define XmNleftWidget "leftWidget"
#define XmNlightThreshold "lightThreshold"
#define XmNlineSpace "lineSpace"
#define XmNlistItemCount "listItemCount"
#define XmNlistItems "listItems"
#define XmNlistLabelString "listLabelString"
#define XmNlistMarginHeight "listMarginHeight"
#define XmNlistMarginWidth "listMarginWidth"
#define XmNlistSizePolicy "listSizePolicy"
#define XmNlistSpacing "listSpacing"
#define XmNlistUpdated "listUpdated"
#define XmNlistVisibleItemCount "listVisibleItemCount"
#define XmNlogicalParent "logicalParent"
#define XmNlosePrimaryCallback "losePrimaryCallback"
#define XmNlosingFocusCallback "losingFocusCallback"
#define XmNmainWindowMarginHeight "mainWindowMarginHeight"
```

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

Page: 340

Figure 6-303: `<Xm/XmStrDefs.h>`***, Part 16 of 34**

```
#define XmNmainWindowMarginWidth "mainWindowMarginWidth"
#define XmNmapCallback "mapCallback"
#define XmNmappingDelay "mappingDelay"
#define XmNmargin "margin"
#define XmNmarginBottom "marginBottom"
#define XmNmarginHeight "marginHeight"
#define XmNmarginLeft "marginLeft"
#define XmNmarginRight "marginRight"
#define XmNmarginTop "marginTop"
#define XmNmarginWidth "marginWidth"
#define XmNmask "mask"
#define XmNmaxLength "maxLength"
#define XmNmaximum "maximum"
#define XmNmenuAccelerator "menuAccelerator"
#define XmNmenuBar "menuBar"
#define XmNmenuCursor "menuCursor"
#define XmNmenuHelpWidget "menuHelpWidget"
#define XmNmenuHistory "menuHistory"
#define XmNmenuPost "menuPost"
#define XmNmessageAlignment "messageAlignment"
#define XmNmessageProc "messageProc"
#define XmNmessageString "messageString"
#define XmNmessageWindow "messageWindow"
#define XmNminimizeButtons "minimizeButtons"
#define XmNminimum "minimum"
#define XmNmnemonic "mnemonic"
#define XmNmnemonicCharSet "mnemonicCharSet"
#define XmNmodifyVerifyCallback "modifyVerifyCallback"
#define XmNmodifyVerifyCallbackWcs "modifyVerifyCallbackWcs"
#define XmNmotionVerifyCallback "motionVerifyCallback"
#define XmNmoveOpaque "moveOpaque"
#define XmNmultiClick "multiClick"
#define XmNmultipleSelectionCallback "multipleSelectionCallback"
#define XmNmustMatch "mustMatch"
#define XmNmwmDecorations "mwmDecorations"
```

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

Page: 341

Figure 6-304: `<Xm/XmStrDefs.h>`*, **Part 17 of 34**

```
#define XmNmwmFunctions "mwmFunctions"
#define XmNmwmInputMode "mwmInputMode"
#define XmNmwmMenu "mwmMenu"
#define XmNmwmMessages "mwmMessages"
#define XmNnavigationType "navigationType"
#define XmNneedsMotion "needsMotion"
#define XmNnoMatchCallback "noMatchCallback"
#define XmNnoMatchString "noMatchString"
#define XmNnoResize "noResize"
#define XmNnoneCursorForeground "noneCursorForeground"
#define XmNnotifyProc "notifyProc"
#define XmNnumColumns "numColumns"
#define XmNnumDropRectangles "numDropRectangles"
#define XmNnumDropTransfers "numDropTransfers"
#define XmNnumExportTargets "numExportTargets"
#define XmNnumImportTargets "numImportTargets"
#define XmNnumRectangles "numRectangles"
#define XmNoffsetX "offsetX"
#define XmNoffsetY "offsetY"
#define XmNokCallback "okCallback"
#define XmNokLabelString "okLabelString"
#define XmNoperationChangedCallback "operationChangedCallback"
#define XmNoperationCursorIcon "operationCursorIcon"
#define XmNoptionLabel "optionLabel"
#define XmNoptionMnemonic "optionMnemonic"
#define XmNoutputCreate "outputCreate"
#define XmNpacking "packing"
#define XmNpageDecrementCallback "pageDecrementCallback"
#define XmNpageIncrement "pageIncrement"
#define XmNpageIncrementCallback "pageIncrementCallback"
#define XmNpaneMaximum "paneMaximum"
#define XmNpaneMinimum "paneMinimum"
#define XmNpattern "pattern"
#define XmNpendingDelete "pendingDelete"
#define XmNpopupEnabled "popupEnabled"
```

System Data Interfaces 6-251

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

Page: 342

Figure 6-305: `<Xm/XmStrDefs.h>`*, **Part 18 of 34**

```
#define XmNpositionIndex "positionIndex"
#define XmNpostFromButton "postFromButton"
#define XmNpostFromCount "postFromCount"
#define XmNpostFromList "postFromList"
#define XmNpreeditType "preeditType"
#define XmNprocessingDirection "processingDirection"
#define XmNpromptString "promptString"
#define XmNprotocolCallback "protocolCallback"
#define XmNpushButtonEnabled "pushButtonEnabled"
#define XmNqualifySearchDataProc "qualifySearchDataProc"
#define XmNradioAlwaysOne "radioAlwaysOne"
#define XmNradioBehavior "radioBehavior"
#define XmNrealizeCallback "realizeCallback"
#define XmNrecomputeSize "recomputeSize"
#define XmNrectangles "rectangles"
#define XmNrefigureMode "refigureMode"
#define XmNrepeatDelay "repeatDelay"
#define XmNresizable "resizable"
#define XmNresizeCallback "resizeCallback"
#define XmNresizeHeight "resizeHeight"
#define XmNresizePolicy "resizePolicy"
#define XmNresizeWidth "resizeWidth"
#define XmNrightAttachment "rightAttachment"
#define XmNrightOffset "rightOffset"
#define XmNrightPosition "rightPosition"
#define XmNrightWidget "rightWidget"
#define XmNrowColumnType "rowColumnType"
#define XmNrows "rows"
#define XmNrubberPositioning "rubberPositioning"
#define XmNsashHeight "sashHeight"
#define XmNsashIndent "sashIndent"
#define XmNsashShadowThickness "sashShadowThickness"
#define XmNsash
```

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"
```

System Data Interfaces 6-253

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

Page: 344

Figure 6-307: `<Xm/XmStrDefs.h>`***, Part 20 of 34**

```
#define XmNsource "source"
#define XmNsourceCursorIcon "sourceCursorIcon"
#define XmNsourceIsExternal "sourceIsExternal"
#define XmNsourcePixmapIcon "sourcePixmapIcon"
#define XmNsourceWidget "sourceWidget"
#define XmNsourceWindow "sourceWindow"
#define XmNspacing "spacing"
#define XmNspotLocation "spotLocation"
#define XmNstartTime "startTime"
#define XmNstateCursorIcon "stateCursorIcon"
#define XmNstringDirection "stringDirection"
#define XmNsubMenuId "subMenuId"
#define XmNsymbolPixmap "symbolPixmap"
#define XmNtearOffMenuActivateCallback "tearOffMenuActivateCallback"
#define XmNtearOffMenuDeactivateCallback "tearOffMenuDeactivateCallback"
#define XmNtearOffModel "tearOffModel"
#define XmNtextAccelerators "textAccelerators"
#define XmNtextColumns "textColumns"
#define XmNtextFontList "textFontList"
#define XmNtextString "textString"
#define XmNtextTranslations "textTranslations"
#define XmNtextValue "textValue"
#define XmNtitleString "titleString"
#define XmNtoBottomCallback "toBottomCallback"
#define XmNtoPositionCallback "toPositionCallback"
#define XmNtoTopCallback "toTopCallback"
#define XmNtopAttachment "topAttachment"
#define XmNtopCharacter "topCharacter"
#define XmNtopItemPosition "topItemPosition"
#define XmNtopLevelEnterCallback "topLevelEnterCallback"
#define XmNtopLevelLeaveCallback "topLevelLeaveCallback"
#define XmNtopOffset "topOffset"
#define XmNtopPosition "topPosition"
#define XmNtopShadowColor "topShadowColor"
#define XmNtopShadowPixmap "topShadowPixmap"
```

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

Page: 345

Figure 6-308: `<Xm/XmStrDefs.h>`*, **Part 21 of 34**

```
#define XmNtopWidget "topWidget"
#define XmNtransferProc "transferProc"
#define XmNtransferStatus "transferStatus"
#define XmNtraversalCallback "traversalCallback"
#define XmNtraversalOn "traversalOn"
#define XmNtraversalType "traversalType"
#define XmNtraverseObscuredCallback "traverseObscuredCallback"
#define XmNtreeUpdateProc "treeUpdateProc"
#define XmNtroughColor "troughColor"
#define XmNunitType "unitType"
#define XmNunmapCallback "unmapCallback"
#define XmNunpostBehavior "unpostBehavior"
#define XmNunselectPixmap "unselectPixmap"
#define XmNupdateSliderSize "updateSliderSize"
#define XmNuseAsyncGeometry "useAsyncGeometry"
#define XmNuserData "userData"
#define XmNvalidCursorForeground "validCursorForeground"
#define XmNvalueChangedCallback "valueChangedCallback"
#define XmNvalueWcs "valueWcs"
#define XmNverifyBell "verifyBell"
#define XmNverticalFontUnit "verticalFontUnit"
#define XmNverticalScrollBar "verticalScrollBar"
#define XmNverticalSpacing "verticalSpacing"
#define XmNvisibleItemCount "visibleItemCount"
#define XmNvisibleWhenOff "visibleWhenOff"
#define XmNvisualPolicy "visualPolicy"
#define XmNwhichButton "whichButton"
#define XmNwordWrap "wordWrap"
#define XmNworkWindow "workWindow"
#define XmRAlignment "Alignment"
#define XmRAnimationMask "AnimationMask"
#define XmRAnimationPixmap "AnimationPixmap"
#define XmRAnimationStyle "AnimationStyle"
#define XmRArrowDirection "ArrowDirection"
#define XmRAtomList "AtomList"
```

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

Page: 346

Figure 6-309: `<Xm/XmStrDefs.h>`*, **Part 22 of 34**

```
#define XmRAttachment "Attachment"
#define XmRAudibleWarning "AudibleWarning"
#define XmRAvailability "Availability"
#define XmRBackgroundPixmap "BackgroundPixmap"
#define XmRBlendModel "BlendModel"
#define XmRBooleanDimension "BooleanDimension"
#define XmRBottomShadowPixmap "BottomShadowPixmap"
#define XmRButtonType "ButtonType"
#define XmRCallbackProc "CallbackProc"
#define XmRChar "Char"
#define XmRCharSetTable "CharSetTable"
#define XmRChildHorizontalAlignment "ChildHorizontalAlignment"
#define XmRChildPlacement "ChildPlacement"
#define XmRChildType "ChildType"
#define XmRChildVerticalAlignment "ChildVerticalAlignment"
#define XmRCommandWindowLocation "CommandWindowLocation"
#define XmRCompoundText "CompoundText"
#define XmRDefaultButtonType "DefaultButtonType"
#define XmRDeleteResponse "DeleteResponse"
#define XmRDialogStyle "DialogStyle"
#define XmRDialogType "DialogType"
#define XmRDoubleClickInterval "DoubleClickInterval"
#define XmRDragInitiatorProtocolStyle "DragInitiatorProtocolStyle"
#define XmRDragReceiverProtocolStyle "DragReceiverProtocolStyle"
#define XmRDropSiteActivity "DropSiteActivity"
#define XmRDropSiteOperations "DropSiteOperations"
#define XmRDropSiteType "DropSiteType"
#define XmRDropTransfers "DropTransfers"
#define XmRExtensionType "ExtensionType"
#define XmRFileTypeMask "FileTypeMask"
#define XmRFontList "FontList"
#define XmRGadgetPixmap "GadgetPixmap"
#define XmRHighlightPixmap "HighlightPixmap"
#define XmRHorizontalDimension "HorizontalDimension"
#define XmRHorizontalInt "HorizontalInt"
```

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

Page: 347

Figure 6-310: `<Xm/XmStrDefs.h>`***, Part 23 of 34**

```
#define XmRHorizontalPosition "HorizontalPosition"
#define XmRIconAttachment "IconAttachment"
#define XmRImportTargets "ImportTargets"
#define XmRIndicatorType "IndicatorType"
#define XmRItemCount "ItemCount"
#define XmRItems "Items"
#define XmRKeySym "KeySym"
#define XmRKeySymTable "KeySymTable"
#define XmRKeyboardFocusPolicy "KeyboardFocusPolicy"
#define XmRLabelType "LabelType"
#define XmRListMarginHeight "ListMarginHeight"
#define XmRListMarginWidth "ListMarginWidth"
#define XmRListSizePolicy "ListSizePolicy"
#define XmRListSpacing "ListSpacing"
#define XmRManBottomShadowPixmap "ManBottomShadowPixmap"
#define XmRManForegroundPixmap "ManForegroundPixmap"
#define XmRManHighlightPixmap "ManHighlightPixmap"
#define XmRManTopShadowPixmap "ManTopShadowPixmap"
#define XmRMenuWidget "MenuWidget"
#define XmRMnemonic "Mnemonic"
#define XmRMultiClick "MultiClick"
#define XmRNavigationType "NavigationType"
#define XmRPacking "Packing"
#define XmRPrimForegroundPixmap "PrimForegroundPixmap"
#define XmRProc "Proc"
#define XmRProcessingDirection "ProcessingDirection"
#define X
```

Figure 6-311: `<Xm/XmStrDefs.h>`***, Part 24 of 34**

```
#define XmRSelectionType "SelectionType"
#define XmRSeparatorType "SeparatorType"
#define XmRShadowType "ShadowType"
#define XmRShellHorizDim "ShellHorizDim"
#define XmRShellHorizPos "ShellHorizPos"
#define XmRShellUnitType "ShellUnitType"
#define XmRShellVertDim "ShellVertDim"
#define XmRShellVertPos "ShellVertPos"
#define XmRSizePolicy "SizePolicy"
#define XmRStringDirection "StringDirection"
#define XmRTearOffModel "TearOffModel"
#define XmRTopShadowPixmap "TopShadowPixmap"
#define XmRTransferStatus "TransferStatus"
#define XmRTraversalType "TraversalType"
#define XmRUnitType "UnitType"
#define XmRUnpostBehavior "UnpostBehavior"
#define XmRValueWcs "ValueWcs"
#define XmRVerticalAlignment "VerticalAlignment"
#define XmRVerticalDimension "VerticalDimension"
#define XmRVerticalInt "VerticalInt"
#define XmRVerticalPosition "VerticalPosition"
#define XmRVirtualBinding "VirtualBinding"
#define XmRVisibleItemCount "VisibleItemCount"
#define XmRVisualPolicy "VisualPolicy"
#define XmRWhichButton "WhichButton"
#define XmRXmBackgroundPixmap "XmBackgroundPixmap"
#define XmRXmString "XmString"
#define XmRXmStringCharSet "XmStringCharSet"
#define XmRXmStringTable "XmStringTable"
#define XmVosfActivate "osfActivate"
#define XmVosfAddMode "osfAddMode"
#define XmVosfBackSpace "osfBackSpace"
#define XmVosfBeginLine "osfBeginLine"
#define XmVosfCancel "osfCancel"
#define XmVosfClear "osfClear"
```

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

Page: 349

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

```
#define XmVosfCopy "osfCopy"
#define XmVosfCut "osfCut"
#define XmVosfDelete "osfDelete"
#define XmVosfDown "osfDown"
#define XmVosfEndLine "osfEndLine"
#define XmVosfHelp "osfHelp"
#define XmVosfInsert "osfInsert"
#define XmVosfLeft "osfLeft"
#define XmVosfMenu "osfMenu"
#define XmVosfMenuBar "osfMenuBar"
#define XmVosfPageDown "osfPageDown"
#define XmVosfPageLeft "osfPageLeft"
#define XmVosfPageRight "osfPageRight"
#define XmVosfPageUp "osfPageUp"
#define XmVosfPaste "osfPaste"
#define XmVosfPrimaryPaste "osfPrimaryPaste"
#define XmVosfQuickPaste "osfQuickPaste"
#define XmVosfRight "osfRight"
#define XmVosfSelect "osfSelect"
#define XmVosfUndo "osfUndo"
#define XmVosfUp "osfUp"
#define XmSFONTLIST_DEFAULT_TAG_STRING "FONTLIST_DEFAULT_TAG_STRING"
#define XmSXmFONTLIST_DEFAULT_TAG_STRING "XmFONTLIST_DEFAULT_TAG_STRING"
#define _XmConst /**/

#define XmSTRING_DEFAULT_CHARSET        XmS
#define XmSTRING_ISO8859_1              "ISO8859-1"
#define XmFONTLIST_DEFAULT_TAG          XmSFONTLIST_DEFAULT_TAG_STRING
#define XmFONTLIST_DEFAULT_TAG_STRING   XmSXmFONTLIST_DEFAULT_TAG_STRING

#define XmVaCASCADEBUTTON              "cascadeButton"
#define XmVaCHECKBUTTON                "checkButton"
#define XmVaDOUBLE_SEPARATOR           "doubleSeparator"
#define XmVaPUSHBUTTON                 "pushButton"
#define XmVaRADIOBUTTON                "radioButton"
```

System Data Interfaces 6-259

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

Page: 350

Figure 6-313: `<Xm/XmStrDefs.h>`*, **Part 26 of 34**

```
#define XmVaSEPARATOR                    "separator"
#define XmVaSINGLE_SEPARATOR             "singleSeparator"
#define XmVaTOGGLEBUTTON                 "checkButton"
#define XmVaTITLE                        XtNtitle

#define XtCKeyboardFocusPolicy           XmCKeyboardFocusPolicy
#define XtCShellUnitType                 XmCShellUnitType
#define XtNkeyboardFocusPolicy           XmNkeyboardFocusPolicy
#define XtNshellUnitType                 XmNshellUnitType
#define XtRKeyboardFocusPolicy           XmRKeyboardFocusPolicy

#define XmRPrimBottomShadowPixmap        XmRBottomShadowPixmap
#define XmRPrimHighlightPixmap           XmRHighlightPixmap
#define XmRPrimTopShadowPixmap           XmRTopShadowPixmap

#define XmCAccelerators                  XtCAccelerators
#define XmCAllowShellResize              XtCAllowShellResize
#define XmCArgc                          XtCArgc
#define XmCArgv                          XtCArgv
#define XmCBackground                    XtCBackground
#define XmCBaseHeight                    XtCBaseHeight
#define XmCBaseHeight                    XtCBaseHeight
#define XmCBaseWidth                     XtCBaseWidth
#define XmCBaseWidth                     XtCBaseWidth
#define XmCBitmap                        XtCBitmap
#define XmCBoolean                       XtCBoolean
#define XmCBorderColor                   XtCBorderColor
#define XmCBorderWidth                   XtCBorderWidth
#define XmCCallback                      XtCCallback
#define XmCColor                         XtCColor
#define XmCColormap                      XtCColormap
#define XmCCreatePopupChildProc          XtCCreatePopupChildProc
#define XmCCursor                        XtCCursor
#define XmCDepth                         XtCDepth
#define XmCDimension                     XtRDimension
```

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

Page: 351

Figure 6-314: `<Xm/XmStrDefs.h>`*, **Part 27 of 34**

```
#define XmCEditMode                     XtREditMode
#define XmCEditType                     XtCEditType
#define XmCEventBindings                XtCEventBindings
#define XmCFile                         XtCFile
#define XmCFont                         XtCFont
#define XmCFontSet                      XtCFontSet
#define XmCForeground                   XtCForeground
#define XmCFraction                     XtCFraction
#define XmCFunction                     XtCFunction
#define XmCGeometry                     XtCGeometry
#define XmCHSpace                       XtCHSpace
#define XmCHeight                       XtCHeight
#define XmCHeightInc                    XtCHeightInc
#define XmCIconMask                     XtCIconMask
#define XmCIconName                     XtCIconName
#define XmCIconNameEncoding             XtCIconNameEncoding
#define XmCIconPixmap                   XtCIconPixmap
#define XmCIconWindow                   XtCIconWindow
#define XmCIconX                        XtCIconX
#define XmCIconY                        XtCIconY
#define XmCIconic                       XtCIconic
#define XmCIndex                        XtCIndex
#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
```

System Data Interfaces 6-261

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

Page: 352

Figure 6-315: `<Xm/XmStrDefs.h>`***, Part 28 of 34**

```
#define XmCMaxWidth                    XtCMaxWidth
#define XmCMenuEntry                   XtCMenuEntry
#define XmCMinAspectX                  XtCMinAspectX
#define XmCMinAspectY                  XtCMinAspectY
#define XmCMinHeight                   XtCMinHeight
#define XmCMinWidth                    XtCMinWidth
#define XmCNotify                      XtCNotify
#define XmCOrientation                 XtCOrientation
#define XmCOverrideRedirect            XtCOverrideRedirect
#define XmCParameter                   XtCParameter
#define XmCPixmap                      XtCPixmap
#define XmCPosition                    XtCPosition
#define XmCReadOnly                    XtCReadOnly
#define XmCResize                      XtCResize
#define XmCReverseVideo                XtCReverseVideo
#define XmCSaveUnder                   XtCSaveUnder
#define XmCScreen                      XtCScreen
#define XmCScrollDCursor               XtCScrollDCursor
#define XmCScrollHCursor               XtCScrollHCursor
#define XmCScrollLCursor               XtCScrollLCursor
#define XmCScrollProc                  XtCScrollProc
#define XmCScrollRCursor               XtCScrollRCursor
#define XmCScrollUCursor               XtCScrollUCursor
#define XmCScrollVCursor               XtCScrollVCursor
#define XmCSelection                   XtCSelection
#define XmCSelectionArray              XtCSelectionArray
#define XmCSensitive                   XtCSensitive
#define XmCSpace                       XtCSpace
#define XmCString                      XtCString
#define XmCTextOptions                 XtCTextOptions
#define XmCTextPosition                XtCTextPosition
#define XmCTextSink                    XtCTextSink
#define XmCTextSource                  XtCTextSource
#define XmCThickness                   XtCThickness
#define XmCThumb                       XtCThumb
```

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

Page: 353

Figure 6-316: `<Xm/XmStrDefs.h>`*, **Part 29 of 34**

```
#define XmCTitle                     XtCTitle
#define XmCTitleEncoding             XtCTitleEncoding
#define XmCTransient                 XtCTransient
#define XmCTransientFor              XtCTransientFor
#define XmCTranslations              XtCTranslations
#define XmCVSpace                    XtCVSpace
#define XmCValue                     XtCValue
#define XmCVisual                    XtCVisual
#define XmCWaitForWm                 XtCWaitForWm
#define XmCWidget                    XtRWidget
#define XmCWidth                     XtCWidth
#define XmCWidthInc                  XtCWidthInc
#define XmCWinGravity                XtCWinGravity
#define XmCWindow                    XtCWindow
#define XmCWindowGroup               XtCWindowGroup
#define XmCWmTimeout                 XtCWmTimeout
#define XmCX                         XtCX
#define XmCY                         XtCY
#define XmNaccelerators              XtNaccelerators
#define XmNallowShellResize          XtNallowShellResize
#define XmNancestorSensitive         XtNancestorSensitive
#define XmNargc                      XtNargc
#define XmNargv                      XtNargv
#define XmNbackground                XtNbackground
#define XmNbackgroundPixmap          XtNbackgroundPixmap
#define XmNbaseHeight                XtNbaseHeight
#define XmNbaseHeight                XtNbaseHeight
#define XmNbaseWidth                 XtNbaseWidth
#define XmNbaseWidth                 XtNbaseWidth
#define XmNbitmap                    XtNbitmap
#define XmNborder                    XtNborder
#define XmNborderColor               XtNborderColor
#define XmNborderPixmap              XtNborderPixmap
#define XmNborderWidth               XtNborderWidth
#define XmNcallback                  XtNcallback
```

Figure 6-317: `<Xm/XmStrDefs.h>`*, **Part 30 of 34**

```
#define XmNchildren                    XtNchildren
#define XmNcolormap                    XtNcolormap
#define XmNcreatePopupChildProc        XtNcreatePopupChildProc
#define XmNdepth                       XtNdepth
#define XmNdestroyCallback             XtNdestroyCallback
#define XmNeditType                    XtNeditType
#define XmNfile                        XtNfile
#define XmNfont                        XtNfont
#define XmNfontSet                     XtNfontSet
#define XmNforceBars                   XtNforceBars
#define XmNforeground                  XtNforeground
#define XmNfunction                    XtNfunction
#define XmNgeometry                    XtNgeometry
#define XmNheight                      XtNheight
#define XmNheightInc                   XtNheightInc
#define XmNhighlight                   XtNhighlight
#define XmNiconMask                    XtNiconMask
#define XmNiconName                    XtNiconName
#define XmNiconNameEncoding            XtNiconNameEncoding
#define XmNiconPixmap                  XtNiconPixmap
#define XmNiconWindow                  XtNiconWindow
#define XmNiconX                       XtNiconX
#define XmNiconY                       XtNiconY
#define XmNiconic                      XtNiconic
#define XmNindex                       XtNindex
#define XmNinitialResourcesPersistent  XtNinitialResourcesPersistent
#define XmNinitialState                XtNinitialState
#define XmNinnerHeight                 XtNinnerHeight
#define XmNinnerWidth                  XtNinnerWidth
#define XmNinnerWindow                 XtNinnerWindow
#define XmNinput                       XtNinput
#define XmNinsertPosition              XtNinsertPosition
#define XmNinternalHeight              XtNinternalHeight
#define XmNinternalWidth               XtNinternalWidth
#define XmNjumpProc                    XtNjumpProc
```

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

Page: 355

Figure 6-318: `<Xm/XmStrDefs.h>`***, Part 31 of 34**

```
#define XmNjustify                 XtNjustify
#define XmNlength                  XtNlength
#define XmNlowerRight              XtNlowerRight
#define XmNmappedWhenManaged       XtNmappedWhenManaged
#define XmNmaxAspectX              XtNmaxAspectX
#define XmNmaxAspectY              XtNmaxAspectY
#define XmNmaxHeight               XtNmaxHeight
#define XmNmaxWidth                XtNmaxWidth
#define XmNmenuEntry               XtNmenuEntry
#define XmNminAspectX              XtNminAspectX
#define XmNminAspectY              XtNminAspectY
#define XmNminHeight               XtNminHeight
#define XmNminWidth                XtNminWidth
#define XmNname                    XtNname
#define XmNnotify                  XtNnotify
#define XmNnumChildren             XtNnumChildren
#define XmNorientation             XtNorientation
#define XmNoverrideRedirect        XtNoverrideRedirect
#define XmNparameter               XtNparameter
#define XmNpixmap                  XtNpixmap
#define XmNpopdownCallback         XtNpopdownCallback
#define XmNpopupCallback           XtNpopupCallback
#define XmNresize                  XtNresize
#define XmNreverseVideo            XtNreverseVideo
#define XmNsaveUnder               XtNsaveUnder
#define XmNscreen                  XtNscreen
#define XmNscrollDCursor           XtNscrollDCursor
#define XmNscrollHCursor           XtNscrollHCursor
#define XmNscrollLCursor           XtNscrollLCursor
#define XmNscrollProc              XtNscrollProc
#define XmNscrollRCursor           XtNscrollRCursor
#define XmNscrollUCursor           XtNscrollUCursor
#define XmNscrollVCursor           XtNscrollVCursor
#define XmNselection               XtNselection
#define XmNselectionArray          XtNselectionArray
```

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

Page: 356

Figure 6-319: `<Xm/XmStrDefs.h>`***, Part 32 of 34**

```
#define XmNsensitive              XtNsensitive
#define XmNshown                  XtNshown
#define XmNspace                  XtNspace
#define XmNstring                 XtNstring
#define XmNtextOptions            XtNtextOptions
#define XmNtextSink               XtNtextSink
#define XmNtextSource             XtNtextSource
#define XmNthickness              XtNthickness
#define XmNthumb                  XtNthumb
#define XmNthumbProc              XtNthumbProc
#define XmNtitle                  XtNtitle
#define XmNtitleEncoding          XtNtitleEncoding
#define XmNtop                    XtNtop
#define XmNtransient              XtNtransient
#define XmNtransientFor           XtNtransientFor
#define XmNtransientFor           XtNtransientFor
#define XmNtranslations           XtNtranslations
#define XmNupdate                 XtNupdate
#define XmNuseBottom              XtNuseBottom
#define XmNuseRight               XtNuseRight
#define XmNvalue                  XtNvalue
#define XmNvisual                 XtNvisual
#define XmNwaitForWm              XtNwaitForWm
#define XmNwidth                  XtNwidth
#define XmNwidthInc               XtNwidthInc
#define XmNwinGravity             XtNwinGravity
#define XmNwindow                 XtNwindow
#define XmNwindowGroup            XtNwindowGroup
#define XmNwmTimeout              XtNwmTimeout
#define XmNx                      XtNx
#define XmNy                      XtNy
#define XmRAcceleratorTable       XtRAcceleratorTable
#define XmRAtom                   XtRAtom
#define XmRBitmap                 XtRBitmap
#define XmRBool                   XtRBool
```

Figure 6-320: `<Xm/XmStrDefs.h>`*, **Part 33 of 34**

```
#define XmRBoolean                    XtRBoolean
#define XmRCallProc                   XtRCallProc
#define XmRCallback                   XtRCallback
#define XmRCardinal                   XtRCardinal
#define XmRColor                      XtRColor
#define XmRColormap                   XtRColormap
#define XmRCursor                     XtRCursor
#define XmRDimension                  XtRDimension
#define XmRDisplay                    XtRDisplay
#define XmREditMode                   XtREditMode
#define XmREnum                       XtREnum
#define XmRFile                       XtRFile
#define XmRFloat                      XtRFloat
#define XmRFont                       XtRFont
#define XmRFontSet                    XtRFontSet
#define XmRFontStruct                 XtRFontStruct
#define XmRFunction                   XtRFunction
#define XmRGeometry                   XtRGeometry
#define XmRImmediate                  XtRImmediate
#define XmRInitialState               XtRInitialState
#define XmRInt                        XtRInt
#define XmRJustify                    XtRJustify
#define XmRLongBoolean                XtRLongBoolean
#define XmROrientation                XtROrientation
#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               XtCTextPosition
```

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

Page: 358

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

```
#define XmRTranslationTable          XtRTranslationTable
#define XmRUnsignedChar              XtRUnsignedChar
#define XmRVisual                    XtRVisual
#define XmRWidget                    XtRWidget
#define XmRWidgetClass               XtRWidgetClass
#define XmRWidgetList                XtRWidgetList
#define XmRWindow                    XtRWindow
```

TCP/IP Data Definitions

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

This section contains standard data definitions that describe system data for the optional TCP/IP Interfaces. These data definitions are referred to by their names in angle brackets: <*name.h*> and <sys/*name.h*>. Included in these data definitions are macro definitions and structure definitions. While an ABI-conforming system may provide TCP/IP interfaces, it need not contain the actual data definitions referenced here. Programmers should observe that the sources of the structures defined in these data definitions are defined in SVID.

Figure 6-322: `<netinet/in.h>`

```
#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
```

M

Figure 6-323: `<netinet/ip.h>`

```
#define      IPOPT_EOL           0
#define      IPOPT_NOP           1
#define      IPOPT_LSRR          131
#define      IPOPT_SSRR          137
```

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

```
#define      TCP_NODELAY   0x01
```

7   DEVELOPMENT ENVIRONMENT

Table of Contents   **i**

Development Commands

The Development Environment for Intel386 implementations of UnixWare® 2.0   M
will contain all of the development commands required by the System V ABI,
namely;

```
as    cc    ld
m4    lex   yacc
```

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.

DRAFT COPY
March 19, 1997
File: abi_386/chap7 (Delta 44.4)
386:adm.book:sum

Page: 364

Software Packaging Tools

The development environment for i386 implementations of the System V ABI shall include each of the following commands as defined in the AS_CMD section of - *System V Interface Definition, Edition 4*.

```
pkgproto    pkgtrans    pkgmk
```

System Headers

Systems that do not have an ABI Development Environment may not have system header files. If an ABI Development Environment is supported, system header files will be included with the Development Environment. The primary source for contents of header files is always the *System V Interface Definition, Edition 4*. In those cases where SVID Fourth Edition doesn't specify the contents of system headers, Chapter 6 "Data Definitions" of this document shall define the associations of data elements to system headers for compilation. For greatest source portability, applications should only depend on header file contents defined in SVID.

Static Archives

Level 1 interfaces defined in *System V Interface Definition, Edition 4*, for each of the following libraries, may be statically linked safely into applications. The resulting executable will not be made non-compliant to the ABI solely because of the static linkage of such members in the executable.

```
libcurses    libm
```

The archive `libcurses.a` is located in /usr/lib on conforming i386 development   M
environments. The archive `libm.a` is located in /usr/lib on conforming i386
development environments.

8 EXECUTION ENVIRONMENT

Table of Contents   **i**

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 `ttyname()` call. Applications must not have the device names of individual terminals hard-coded within them. The `sxt` entries are optional in the system but, if present must be included in the library routine's search.

The following device files are required to be present on all i386 ABI-conforming computers that support the corresponding hardware devices.

DRAFT COPY
March 19, 1997
File: abi_386/chap8 (Delta 44.5)
386:adm.book:sum

Page: 367

`/dev/lpX`	This device file is the lineprinter device. The letter ''X'' represents a one-digit integer that identifies the particular lineprinter device.
`/dev/dsk/`	
`/dev/rdsk/`	These directories contain the raw and block disk device files. They are of the form:

        `f[01][t]`
        `f[01][35][dh][t]`
        `c#t#d#s#`

where 'c' is followed by a controller number,
   't' is followed by a target number,
   'd' is followed by a disk unit number,
   's' is followed by a disk slice number.

`/dev/rmt/`	These directories contain the raw and block tape device files. The devices guaranteed to be in this directory are:

        `ctape1`
        `ntape1`

`/dev/cdrom/`	
`/dev/rcdrom/`	These directories contain the raw and block CD-ROM disk device files. They are of the form:

        `c#t#l#`
        `c#t#l#`
        `cdrom#`

The letter 'c' is followed by a controller number. The letter 't' is followed by a target number on the controller. The letter 'l' is followed by a logical unit number on the target. The device "cdrom" is followed by a sequential number as nodes are created.

No leading zeroes are used in the numbers (target four is t4 not t04). The numbering for 'c', 't' and 'l' begins at zero and the numbering for 'n' begins at one.

IN Index

Index

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
architecture

implementation   3: 1
  processor   3: 1
archive file   7: 2
argc   3: 26
arguments
  bad assumptions   3: 44
  exec(BA_OS)   3: 26
  floating-point   3: 17
  function   3: 9
  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

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

Page: 372

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
priocntl.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

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

Page: 374

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

Page: 376

V

W

X

Y

Z