

Linux Standard Base Core Specification

2.0.1

|

Linux Standard Base Core Specification 2.0.1

Copyright © 2004 Free Standards Group

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Portions of the text are copyrighted by the following parties:

- The Regents of the University of California
- Free Software Foundation
- Ian F. Darwin
- Paul Vixie
- BSDI (now Wind River)
- Andrew G Morgan
- Jean-loup Gailly and Mark Adler
- Massachusetts Institute of Technology

These excerpts are being used in accordance with their respective licenses.

Linux is a trademark of Linus Torvalds.

UNIX a registered trademark of the Open Group in the United States and other countries.

LSB is a trademark of the Free Standards Group in the USA and other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

Intel and Itanium are registered trademarks and Intel386 is a trademarks of Intel Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Specification Introduction

Specification Introduction

Table of Contents

Foreword	i
Introduction	ii
I. Introductory Elements	3
1. Scope	1
1.1. General	1
1.2. Module Specific Scope	1
2. Normative References	2
3. Requirements	6
3.1. Relevant Libraries	6
3.2. LSB Implementation Conformance	6
3.3. LSB Application Conformance	7
4. Definitions	8
5. Terminology	9
6. Documentation Conventions	10

List of Tables

2-1. Normative References2

3-1. Standard Library Names.....6

3-2. Standard Library Names defined in the Architecture Specific Supplement6

Foreword

- 1 | This is version 2.0.1 of the Linux Standard Base Core Specification. An implementation of this version of the
- 2 | specification may not claim to be an implementation of the Linux Standard Base unless it has successfully completed
- 3 | the compliance process as defined by the Free Standards Group.

Introduction

1 The LSB defines a binary interface for application programs that are compiled and packaged for LSB-conforming
2 implementations on many different hardware architectures. Since a binary specification shall include information
3 specific to the computer processor architecture for which it is intended, it is not possible for a single document to
4 specify the interface for all possible LSB-conforming implementations. Therefore, the LSB is a family of
5 specifications, rather than a single one.

6 This document should be used in conjunction with the documents it references. This document enumerates the system
7 components it includes, but descriptions of those components may be included entirely or partly in this document,
8 partly in other documents, or entirely in other reference documents. For example, the section that describes system
9 service routines includes a list of the system routines supported in this interface, formal declarations of the data
10 structures they use that are visible to applications, and a pointer to the underlying referenced specification for
11 information about the syntax and semantics of each call. Only those routines not described in standards referenced by
12 this document, or extensions to those standards, are described in the detail. Information referenced in this way is as
13 much a part of this document as is the information explicitly included here.

I. Introductory Elements

Chapter 1. Scope

1.1. General

1 The Linux Standard Base (LSB) defines a system interface for compiled applications and a minimal environment for
2 support of installation scripts. Its purpose is to enable a uniform industry standard environment for high-volume
3 applications conforming to the LSB.

4 These specifications are composed of two basic parts: A common specification ("LSB-generic") describing those parts
5 of the interface that remain constant across all implementations of the LSB, and an architecture-specific specification
6 ("LSB-arch") describing the parts of the interface that vary by processor architecture. Together, the LSB-generic and
7 the architecture-specific supplement for a single hardware architecture provide a complete interface specification for
8 compiled application programs on systems that share a common hardware architecture.

9 The LSB-generic document shall be used in conjunction with an architecture-specific supplement. Whenever a section
10 of the LSB-generic specification shall be supplemented by architecture-specific information, the LSB-generic
11 document includes a reference to the architecture supplement. Architecture supplements may also contain additional
12 information that is not referenced in the LSB-generic document.

13 The LSB contains both a set of Application Program Interfaces (APIs) and Application Binary Interfaces (ABIs). APIs
14 may appear in the source code of portable applications, while the compiled binary of that application may use the
15 larger set of ABIs. A conforming implementation shall provide all of the ABIs listed here. The compilation system
16 may replace (e.g. by macro definition) certain APIs with calls to one or more of the underlying binary interfaces, and
17 may insert calls to binary interfaces as needed.

18 The LSB is primarily a binary interface definition. Not all of the source level APIs available to applications may be
19 contained in this specification.

1.2. Module Specific Scope

20 This is the Core module of the Linux Standards Base (LSB). This module provides the fundamental system interfaces,
21 libraries, and runtime environment upon which all conforming applications and libraries depend.

22 Interfaces described in this module are mandatory except where explicitly listed otherwise. Core interfaces may be
23 supplemented by other modules; all modules are built upon the core.

Chapter 2. Normative References

The specifications listed below are referenced in whole or in part by the Linux Standard Base. In this specification, where only a particular section of one of these references is identified, then the normative reference is to that section alone, and the rest of the referenced document is informative.

Table 2-1. Normative References

System V Application Binary Interface—DRAFT—17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://www.eagereon.com/dwarf/dwarf-2.0.0.pdf
Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEEE Standard 754 for Binary Floating-Point Arithmetic	http://www.ieee.org/
System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspees/gabi41.pdf
ISO/IEC 9899: 1999, Programming Language—C	
Linux Assigned Names And Numbers Authority	http://www.lanana.org/
Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
LI18NIX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NIX-2000-amd4.htm
Linux Standard Base	http://www.linuxbase.org/spec/
OSF RFC 86.0	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt
RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	http://www.opengroup.org/publications/catalog/un.htm
The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/publications/catalog/un.htm
CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,	http://www.opengroup.org/publications/catalog/un.htm

C606)		
ISO/IEC 9945:2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3		http://www.unix.org/version3/
System V Interface Definition, Issue 3 (ISBN 0201566524)		
System V Interface Definition, Fourth Edition		
zlib 1.2 Manual		http://www.gzip.org/zlib/
Name	Title	URL
DWARF Debugging Information Format	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://www.eagercon.com/dwarf/dwarf-2.0.0.pdf
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEEE Std 754-1985	IEEE Standard 754 for Binary Floating-Point Arithmetic	http://www.ieee.org/
ISO C (1999)	ISO/IEC 9899: 1999, Programming Language --C	
ISO POSIX (2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale	http://www.unix.org/version3/
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
Li18nux Globalization Specification	LI18NUNIX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUNIX-2000-amd4.htm

Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt
PAM	Open Software Foundation, Request For Comments: 86.0 , October 1995, V. Samar & R.Schemers (SunSoft)	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
SUSv2	CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)	http://www.opengroup.org/publications/catalog/un.htm
SUSv2 Command and Utilities	The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/publications/catalog/un.htm
SVID Issue 3	American Telephone and Telegraph Company, System V Interface Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989.(ISBN 0201566524)	
SVID Issue 4	System V Interface Definition,Fourth Edition	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
this specification	Linux Standard Base	http://www.linuxbase.org/spec/
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2	http://www.opengroup.org/publications/catalog/un.htm

5

	(ISBN: 1-85912-171-3, C610), plus Corrigendum U018	ons/catalog/un.htm
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

Chapter 3. Requirements

3.1. Relevant Libraries

The libraries listed in Table 3-1 shall be available on a Linux Standard Base system, with the specified runtime names. The libraries listed in Table 3-2 are architecture specific, but shall be available on all LSB conforming systems. This list may be supplemented or amended by the architecture-specific specification.

Table 3-1. Standard Library Names

Library	Runtime Name
libdl	libdl.so.2
libcrypt	libcrypt.so.1
libzlibdl	libzlibdl.so.42
libncurses	libncurses.so.5
libutil	libutil.so.1
libpthread	libpthread.so.0
libutil	libutil.so.1
libz	libz.so.1
libpam	libpam.so.0
libgcc_s	libgcc_s.so.1

Table 3-2. Standard Library Names defined in the Architecture Specific Supplement

Library	Runtime Name
libc	See archLSB
libm	See archLSB
libe	See archLSB
proginterp	See archLSB

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

3.2. LSB Implementation Conformance

An A conforming implementation shall satisfy the following requirements:

- The implementation shall implement fully the architecture described in the hardware manual for the target processor architecture.

- The implementation shall be capable of executing compiled applications having the format and using the system interfaces described in this document.
- The implementation shall provide libraries containing the interfaces specified by this document, and shall provide a dynamic linking mechanism that allows these interfaces to be attached to applications at runtime. All the interfaces shall behave as specified in this document.
- The map of virtual memory provided by the implementation shall conform to the requirements of this document.
- The implementation's low-level behavior with respect to function call linkage, system traps, signals, and other such activities shall conform to the formats described in this document.
- The implementation shall provide all of the mandatory interfaces in their entirety.
- The implementation may provide one or more of the optional interfaces. Each optional interface that is provided shall be provided in its entirety. The product documentation shall state which optional interfaces are provided.
- The implementation shall provide all files and utilities specified as part of this document in the format defined here and in other referenced documents. All commands and utilities shall behave as required by this document. The implementation shall also provide all mandatory components of an application's runtime environment that are included or referenced in this document.
- The implementation, when provided with standard data formats and values at a named interface, shall provide the behavior defined for those values and data formats at that interface. However, a conforming implementation may consist of components which are separately packaged and/or sold. For example, a vendor of a conforming implementation might sell the hardware, operating system, and windowing system as separately packaged items.
- The implementation may provide additional interfaces with different names. It may also provide additional behavior corresponding to data values outside the standard ranges, for standard named interfaces.

3.3. LSB Application Conformance

A conforming application shall satisfy the following requirements:

- Its executable files are either shell scripts or object files in the format defined for the Object File Format system interface.
- Its object files participate in dynamic linking as defined in the Program Loading and Linking System interface.
- It employs only the instructions, traps, and other low-level facilities defined in the Low-Level System interface as being for use by applications.
- If it requires any optional interface defined in this document in order to be installed or to execute successfully, the requirement for that optional interface is stated in the application's documentation.
- It does not use any interface or data format that is not required to be provided by a conforming implementation, unless:
 - If such an interface or data format is supplied by another application through direct invocation of that application during execution, that application is in turn an LSB conforming application.
 - The use of that interface or data format, as well as its source, is identified in the documentation of the application.
- It shall not use any values for a named interface that are reserved for vendor extensions.

A strictly conforming application does not require or use any interface, facility, or implementation-defined extension that is not defined in this document in order to be installed or to execute successfully.

Chapter 4. Definitions

1 For the purposes of this document, the following definitions, as specified in the *ISO/IEC Directives, Part 2, 2001, 4th*
2 *Edition*, apply:

3 can

4 be able to; there is a possibility of; it is possible to

5 cannot

6 be unable to; there is no possibility of; it is not possible to

7 may

8 is permitted; is allowed; is permissible

9 need not

10 it is not required that; no...is required

11 shall

12 is to; is required to; it is required that; has to; only...is permitted; it is necessary

13 shall not

14 is not allowed [permitted] [acceptable] [permissible]; is required to be not; is required that...be not; is not to be

15 should

16 it is recommended that; ought to

17 should not

18 it is not recommended that; ought not to

Chapter 5. Terminology

For the purposes of this document, the following terms apply:

archLSB

The architectural part of the LSB Specification which describes the specific parts of the interface that are platform specific. The archLSB is complementary to the gLSB.

Binary Standard

The total set of interfaces that are available to be used in the compiled binary code of a conforming application.

gLSB

The common part of the LSB Specification that describes those parts of the interface that remain constant across all hardware implementations of the LSB.

implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

Source Standard

The set of interfaces that are available to be used in the source code of a conforming application.

undefined

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

Other terms and definitions used in this document shall have the same meaning as defined in Chapter 3 of the Base Definitions volume of ISO POSIX (2003).

Chapter 6. Documentation Conventions

Throughout this document, the following typographic conventions are used:

`function()`

the name of a function

command

the name of a command or utility

CONSTANT

a constant value

parameter

a parameter

variable

a variable

Throughout this specification, several tables of interfaces are presented. Each entry in these tables has the following format:

name

the name of the interface

(symver)

An optional symbol version identifier, if required.

[*refno*]

A reference number indexing the table of referenced specifications that follows this table.

For example,

forkpty(GLIBC_2.0) [1]

refers to the interface named `forkpty` with symbol version `GLIBC_2.0` that is defined in the first of the listed references below the table.

ELF Specification

2

3 **ELF Specification**

Table of Contents

I. Low Level System Information	16
1. Operating System Interface	1
II. Object Format	2
2. Object Files	3
3. Sections	4
3.1. Sections Types	4
3.1.1. ELF Section Types	4
3.1.2. Additional Section Types	7
4. Special Sections	8
4.1. Special Sections	8
4.1.1. ELF Special Sections	8
4.1.2. Additional Special Sections	11
5. Symbol Mapping	13
5.1. Symbol Mapping	13
5.1.1. C Language	13
5.1.2. C++ Language	14
6. DWARF Extensions	14
7. EH Frame Header	16
7.1. DWARF Exception Header Encoding	17
8. Symbol Versioning	18
8.1. Symbol Versioning	18
8.1.1. Version Table Definitions	18
8.2.3. Version Definitions Requirements	19
8.3. Version Requirements	19
8.4. Startup Sequence	21
8.5. Symbol Resolution	21
9. ABI note tag	22
III. Dynamic Linking	23
10. Program Loading and Dynamic Linking	24
11. Program Loading and Dynamic Linking Header	25
12. Dynamic Entries	26
12.1. Dynamic Entries	26
12.1.1. ELF Dynamic Entries	26
12.1.1. ELF2. Additional Dynamic Entries	28
12.1.2. Additional Dynamic Entries	000

List of Tables

3-1. ELF Section Types	4
3-2. Additional Section Types	7
4-1. ELF Special Sections.....	8
4-2. Additional Special Sections.....	11
6-1. Additional DWARF Call Frame Instructions	14
7-1. .eh_frame_hdr Section Format.....	16
7-2. DWARF Exception Header value format.....	17
7-3. DWARF Exception Header application	17
11-1. Linux Segment Types.....	25

List of Figures

8-1. Version Definition Entries.....	18
8-2. Version Definition Auxiliary Entries.....	19
8-3. Version Needed Entries	20
8-4. Version Needed Auxiliary Entries.....	20

I. Low Level System Information

Chapter 1. Operating System Interface

- 1 LSB-conforming applications shall assume that stack, heap and other allocated memory regions will be
- 2 non-executable. The application must take steps to make them executable if needed.

II. Object Format

Chapter 2. Object Files

LSB-conforming implementations shall support the object file Executable and Linking Format (ELF), which is defined by the following documents:

- System V ~~Application Binary Interface, Edition 4.1~~ABI
- System V ~~Application Binary Interface — DRAFT — 17 December 2003~~ABI Update
- this document
- an architecture-specific LSB specification

Conforming implementations may also support other unspecified object file formats.

Chapter 3. Sections

As described in System V ABI, an ELF object file contains a number of *sections*.

3.1. Sections Types

The section header table is an array of Elf32_Shdr or Elf64_Shdr structures as described in System V ABI. The *sh_type* member shall be either a value from Table 3-1, drawn from the System V ABI, or one of the additional values specified in Table 3-2.

A section header's *sh_type* member specifies the sections's semantics.

3.1.1. ELF Section Types

The following section types are defined in the System V Application Binary Interface, Edition 4.1 ABI and the System V Application Binary Interface DRAFT 17 December 2003 ABI Update.

Table 3-1. ELF Section Types

Name	Value	Description
SHT_DYNAMIC	0x6	The section holds information for dynamic linking. Currently, an object file shall have only one dynamic section, but this restriction may be relaxed in the future. See 'Dynamic Section' in Chapter 5 for details.
SHT_DYNSYM	0xb	This section holds a minimal set of symbols adequate for dynamic linking. See also SHT_SYMTAB. Currently, an object file may have either a section of SHT_SYMTAB type or a section of SHT_DYNSYM type, but not both. This restriction may be relaxed in the future.
SHT_FINI_ARRAY	0xf	This section contains an array of pointers to termination functions, as described in 'Initialization and Termination Functions' in Chapter 5. Each pointer in the array is taken as a parameterless procedure with a void return.
SHT_HASH	0x5	The section holds a symbol hash table. Currently, an object file shall have only one hash table, but this

Name	Value	Description
		restriction may be relaxed in the future. See 'Hash Table' in the Chapter 5 for details.
SHT_HIPROC	0x7fffffff	Values in this inclusive range are reserved for processor-specific semantics.
SHT_HIUSER	0xffffffff	This value specifies the upper bound of the range of indexes reserved for application programs. Section types between SHT_LOUSER and SHT_HIUSER can be used by the application, without conflicting with current or future system-defined section types.
SHT_INIT_ARRAY	0xe	This section contains an array of pointers to initialization functions, as described in 'Initialization and Termination Functions' in Chapter 5. Each pointer in the array is taken as a parameterless procedure with a void return.
SHT_LOPROC	0x70000000	Values in this inclusive range are reserved for processor-specific semantics.
SHT_LOUSER	0x80000000	This value specifies the lower bound of the range of indexes reserved for application programs.
SHT_NOBITS	0x8	A section of this type occupies no space in the file but otherwise resembles SHT_PROGBITS. Although this section contains no bytes, the sh_offset member contains the conceptual file offset.
SHT_NOTE	0x7	The section holds information that marks the file in some way. See 'Note Section' in Chapter 5 for details.
SHT_NULL	0x0	This value marks the section header as inactive; it does not have an associated section. Other members of the section header have undefined

Name	Value	Description
		values.
SHT_PREINIT_ARRAY	0x10	This section contains an array of pointers to functions that are invoked before all other initialization functions, as described in 'Initialization and Termination Functions' in Chapter 5. Each pointer in the array is taken as a parameterless procedure with a void return.
SHT_PROGBITS	0x1	The section holds information defined by the program, whose format and meaning are determined solely by the program.
SHT_REL	0x9	The section holds relocation entries without explicit addends, such as type Elf32_Rel for the 32-bit class of object files or type Elf64_Rel for the 64-bit class of object files. An object file may have multiple relocation sections. See "Relocation"
SHT_RELA	0x4	The section holds relocation entries with explicit addends, such as type Elf32_Rela for the 32-bit class of object files or type Elf64_Rela for the 64-bit class of object files. An object file may have multiple relocation sections. 'Relocation' b
SHT_SHLIB	0xa	This section type is reserved but has unspecified semantics.
SHT_STRTAB	0x3	The section holds a string table. An object file may have multiple string table sections. See 'String Table' below for details.
SHT_SYMTAB	0x2	These sections hold This section holds a symbol table. Currently, an object file shall may have only one either a section of each SHT_SYMTAB type or a section of SHT_DYNSYM type, but this is not both. This restriction may be relaxed in the future. Typically, SHT_SYMTAB provides symbols

Name	Value	Description
		for link editing, though it may also be used for dynamic linking. As a complete symbol table, it may contain many symbols unnecessary for dynamic linking.

3.1.2. Additional Section Types

The following additional section types are defined here.

Table 3-2. Additional Section Types

Name	Value	Description
SHT_GNU_verdef	0x6ffffffd	This section contains the symbol versions that are provided.
SHT_GNU_verneed	0x6ffffffe	This section contains the symbol versions that are required.
SHT_GNU_versym	0x6fffffff	This section contains the Symbol Version Table.

Chapter 4. Special Sections

4.1. Special Sections

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

4.1.1. ELF Special Sections

- 3 The following sections are defined in the System V Application Binary Interface, Edition 4.1 ABI and the System V
4 Application Binary Interface—DRAFT—17 December 2003 ABI Update.

5 **Table 4-1. ELF Special Sections**

Name	Type	Attributes
.bss	SHT_NOBITS	SHF_ALLOC+SHF_WRITE
.comment	SHT_PROGBITS	0
.data	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
.data1	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
.debug	SHT_PROGBITS	0
.dynamic	SHT_DYNAMIC	SHF_ALLOC+SHF_WRITE
.dynstr	SHT_STRTAB	SHF_ALLOC
.dynsym	SHT_DYNSYM	SHF_ALLOC
.fini	SHT_PROGBITS	SHF_ALLOC+SHF_EXECINSTR
.fini_array	SHT_FINI_ARRAY	SHF_ALLOC+SHF_WRITE
.hash	SHT_HASH	SHF_ALLOC
.init	SHT_PROGBITS	SHF_ALLOC+SHF_EXECINSTR
.init_array	SHT_INIT_ARRAY	SHF_ALLOC+SHF_WRITE
.interp	SHT_PROGBITS	SHF_ALLOC
.line	SHT_PROGBITS	0
.note	SHT_NOTE	0
.preinit_array	SHT_PREINIT_ARRAY	SHF_ALLOC+SHF_WRITE
.rodata	SHT_PROGBITS	SHF_ALLOC
.rodata1	SHT_PROGBITS	SHF_ALLOC

Name	Type	Attributes
.shstrtab	SHT_STRTAB	0
.strtab	SHT_STRTAB	SHF_ALLOC
.symtab	SHT_SYMTAB	SHF_ALLOC
.text	SHT_PROGBITS	SHF_ALLOC+SHF_EXECINSTR

.bss

This section holds data that contributes to the program's memory image. The program may treat this data as uninitialized. However, the system shall initialize this data with zeroes when the program begins to run. The section occupies no file space, as indicated by the section type, SHT_NOBITS

.comment

This section holds version control information.

.data

This section holds initialized data that contribute to the program's memory image.

.data1

This section holds initialized data that contribute to the program's memory image.

.debug

This section holds information for symbolic debugging. The contents are unspecified. All section names with the prefix .debug hold information for symbolic debugging. The contents of these sections are unspecified.

.dynamic

This section holds dynamic linking information. The section's attributes will include the SHF_ALLOC bit. Whether the SHF_WRITE bit is set is processor specific. See Chapter 5 for more information.

.dynstr

This section holds strings needed for dynamic linking, most commonly the strings that represent the names associated with symbol table entries. See Chapter 5 for more information.

.dynsym

This section holds the dynamic linking symbol table, as described in 'Symbol Table'. See Chapter 5 for more information.

.fini

This section holds executable instructions that contribute to the process termination code. That is, when a program exits normally, the system arranges to execute the code in this section.

.fini_array

This section holds an array of function pointers that contributes to a single termination array for the executable or shared object containing the section.

35 `.hash`

36 This section holds a symbol hash table. See 'Hash Table' in Chapter 5 for more information.

37 `.init`

38 This section holds executable instructions that contribute to the process initialization code. When a program
39 starts to run, the system arranges to execute the code in this section before calling the main program entry point
40 (called `main` for C programs)

41 `.init_array`

42 This section holds an array of function pointers that contributes to a single initialization array for the executable
43 or shared object containing the section.

44 `.interp`

45 This section holds the path name of a program interpreter. If the file has a loadable segment that includes
46 relocation, the sections' attributes will include the `SHF_ALLOC` bit; otherwise, that bit will be off. See Chapter 5
47 for more information.

48 `.line`

49 This section holds line number information for symbolic debugging, which describes the correspondence
50 between the source program and the machine code. The contents are unspecified.

51 `.note`

52 This section holds information in the format that 'Note Section' in Chapter 5 describes of the System V
53 Application Binary Interface, Edition 4.1.

54 `.preinit_array`

55 This section holds an array of function pointers that contributes to a single pre-initialization array for the
56 executable or shared object containing the section.

57 `.rodata`

58 This section holds read-only data that typically contribute to a non-writable segment in the process image. See
59 'Program Header' in Chapter 5 for more information.

60 `.rodata1`

61 This section hold sread-only data that typically contribute to a non-writable segment in the process image. See
62 'Program Header' in Chapter 5 for more information.

63 `.shstrtab`

64 This section holds section names.

65 `.strtab`

66 This section holds strings, most commonly the strings that represent the names associated with symbol table
67 entries. If the file has a loadable segment that includes the symbol string table, the section's attributes will include
68 the `SHF_ALLOC` bit; otherwi

69 `.symtab`
 70 This section holds a symbol table, as 'Symbol Table' in this chapter describes. If the file has a loadable segment
 71 that includes the symbol table, the section's attributes will include the SHF_ALLOC bit; otherwise, that bit will
 72 be off.

73 `.text`

74 This section holds the 'text,' or executable instructions, of a program.

4.1.2. Additional Special Sections

75 Object files in an LSB conforming application may also contain one or more of the additional special sections
 76 described below.

77 **Table 4-2. Additional Special Sections**

Name	Type	Attributes
<code>.ctors</code>	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
<code>.dtors</code>	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
<code>.eh_frame</code>	SHT_PROGBITS	SHF_ALLOC
<code>.eh_frame_hdr</code>	SHT_PROGBITS	SHF_ALLOC
<code>.gnu.version</code>	SHT_GNU_versym	SHF_ALLOC
<code>.gnu.version_d</code>	SHT_GNU_verdef	SHF_ALLOC
<code>.gnu.version_r</code>	SHT_GNU_verneed	SHF_ALLOC
<code>.jcr</code>	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
<code>.note.ABI-tag</code>	SHT_NOTE	SHF_ALLOC
<code>.stab</code>	SHT_PROGBITS	0
<code>.stabstr</code>	SHT_STRTAB	0

79 `.ctors`

80 This section contains a list of global constructor function pointers.

81 `.dtors`

82 This section contains a list of global destructor function pointers.

83 `.eh_frame`

84 This section contains information necessary for frame unwinding during exception handling.

85 `.eh_frame_hdr`

86 This section contains a pointer to the `.eh_frame` section which is accessible to the runtime support code of a C++
 87 application. This section may also contain a binary search table which may be used by the runtime support code
 88 to more efficiently access records in the `.eh_frame` section.

89 .gnu.version
90 This section contains the Symbol Version Table.

91 .gnu.version_d
92 This section contains the Version Definitions.

93 .gnu.version_r
94 This section contains the Version Requirements.

95 .jcr
96 This section contains information necessary for registering compiled Java classes. The contents are
97 compiler-specific and used by compiler initialization functions.

98 .note.ABI-tag
99 Specify ABI details.

100 .stab
101 This section contains debugging information. The contents are not specified as part of the LSB.

102 .stabstr
103 This section contains strings associated with the debugging information contained in the .stab section.

Chapter 5. Symbol Mapping

- 1 This chapter defines how names are mapped from the source symbol to the object symbol.

5.1. Symbol Mapping

- 2 Symbols in a source program are translated by the compilation system into symbols that exist in the object file. The
3 rules for this translation are defined here.

5.1.1. C Language

- 4 External C symbols have the same names in C and object files' symbol tables.

~~5.1.2. C++ Language~~

- 5 ~~External symbol names in a C++ object file shall be encoded according to the "name mangling" rules described in the.~~

Chapter 6. DWARF Extensions

In addition to the Call Frame Instructions defined in section 6.4.2 of DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993), the following Call Frame Instructions may also be used.

Table 6-1. Additional DWARF Call Frame Instructions

Name	Value	Meaning
DW_CFA_expression	0x10	The DW_CFA_expression instruction takes two operands: an unsigned LEB128 value representing a register number, and a DW_FORM_block value representing a DWARF expression. The required action is to establish the DWARF expression as the means by which the address in which the given register contents are found may be computed. The value of the CFA is pushed on the DWARF evaluation stack prior to execution of the DWARF expression. The DW_OP_call2, DW_OP_call4, DW_OP_call_ref and DW_OP_push_object_address DWARF operators (see Section 2.4.1 of DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)) cannot be used in such a DWARF expression.
DW_CFA_offset_extended_sf	0x11	The DW_CFA_offset_extended_sf instruction takes two operands: an unsigned LEB128 value representing a register number and a signed LEB128 factored offset. This instruction is identical to DW_CFA_offset_extended except that the second operand is signed.
DW_CFA_def_cfa_sf	0x12	The DW_CFA_def_cfa_sf instruction takes two operands: an unsigned LEB128 value representing a register number and a signed LEB128 factored offset. This instruction is identical to

Name	Value	Meaning
		DW_CFA_def_cfa except that the second operand is signed and factored.
DW_CFA_def_cfa_offset_sf	0x13	The DW_CFA_def_cfa_offset_sf instruction takes a signed LEB128 operand representing a factored offset. This instruction is identical to DW_CFA_def_cfa_offset except that the operand is signed and factored.
DW_CFA_GNU_args_size	0x2e	The DW_CFA_def_cfa_offset_sf instruction takes an unsigned LEB128 operand representing an argument size.
DW_CFA_GNU_negative_offset_extended	0x2f	The DW_CFA_def_cfa_sf instruction takes two operands: an unsigned LEB128 value representing a register number and an unsigned LEB128 which represents the magnitude of the offset. This instruction is identical to DW_CFA_offset_extended_sf except that the operand is subtracted to produce the offset. This instructions is obsoleted by DW_CFA_offset_extended_sf.

Chapter 7. EH Frame Header

The `.eh_frame_hdr` section contains additional information about the `.eh_frame` section. A pointer to the start of the `.eh_frame` data, and optionally, a binary search table of pointers to the `.eh_frame` records are found in this section.

Data in this section is encoded according to the DWARF Exception Header Encoding described below.

Table 7-1. `.eh_frame_hdr` Section Format

Encoding	Field
unsigned byte	version
unsigned byte	eh_frame_ptr_enc
unsigned byte	fde_count_enc
unsigned byte	table_enc
encoded	eh_frame_ptr
encoded	fde_count
	binary search table

version

Version of the `.eh_frame_hdr` format. This value shall be 1.

eh_frame_ptr_enc

The encoding format of the `eh_frame_ptr` field.

fde_count_enc

The encoding format of the `fde_count` field. A value of `DW_EH_PE_omit` indicates the binary search table is not present.

table_enc

The encoding format of the entries in the binary search table. A value of `DW_EH_PE_omit` indicates the binary search table is not present.

eh_frame_ptr

The encoded value of the pointer to the start of the `.eh_frame` section.

fde_count

The encoded value of the count of entries in the binary search table.

21 binary search table

22 A binary search table containing fde_count entries. Each entry of the table consist of two encoded values, the
23 initial location, and the address. The entries are sorted in an increasing order by the initial location value.

7.1. DWARF Exception Header Encoding

24 The DWARF Exception Header Encoding is used to describe the type of data used in the .eh_frame_hdr section.
25 The upper 4 bits indicate how the value is to be applied. The lower 4 bits indicate the format of the data.

26 **Table 7-2. DWARF Exception Header value format**

Name	Value	Meaning
DW_EH_PE_omit	0xff	No value is present.
DW_EH_PE_uleb128	0x01	Unsigned value is encoded using the Little Endian Base 128 (LEB128) as defined by DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993).
DW_EH_PE_udata2	0x02	A 2 bytes unsigned value.
DW_EH_PE_udata4	0x03	A 4 bytes unsigned value.
DW_EH_PE_udata8	0x04	An 8 bytes unsigned value.
DW_EH_PE_sleb128	0x09	Signed value is encoded using the Little Endian Base 128 (LEB128) as defined by DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993).
DW_EH_PE_sdata2	0x0A	A 2 bytes signed value.
DW_EH_PE_sdata4	0x0B	A 4 bytes signed value.
DW_EH_PE_sdata8	0x0C	An 8 bytes signed value.

28 **Table 7-3. DWARF Exception Header application**

Name	Value	Meaning
DW_EH_PE_absptr	0x00	Value is used with no modification.
DW_EH_PE_pcrel	0x10	Value is relative to the current program counter.
DW_EH_PE_datarel	0x30	Value is relative to the beginning of the .eh_frame_hdr section.
DW_EH_PE_omit	0xff	No value is present.

Chapter 8. Symbol Versioning

This chapter describes the Symbol Versioning mechanism. All ELF objects may provide or depend on versioned symbols. Symbol Versioning is implemented by 3 section types: SHT_GNU_versym, SHT_GNU_verdef, and SHT_GNU_verneed.

The prefix Elfxx in the following descriptions and code fragments stands for either "Elf32" or "Elf64", depending on the architecture.

Versions are described by strings. The structures that are used for symbol versions also contain a member that holds the ELF hashing values of the strings. This allows for more efficient processing.

8.1. Symbol Version Table

The Symbol Version Table is contained in the special section `.gnu.version` which has a section type of SHT_GNU_versym. This section has the same number of entries as the Dynamic Symbol Table.

This section contains an array of elements of type Elfxx_Half. Each entry specifies the version defined for or required by the corresponding symbol in the Dynamic Symbol Table.

The values in the Symbol Version Table are unique to the object in which they are located. These values are identifiers that are provided by the `vna_other` member of the Elfxx_Vernaux structure or the `vd_ndx` member of the Elfxx_Verdef structure.

The values 0 and 1 are reserved.

0

The symbol is local, not available outside the object.

1

The symbol is defined in this object and is globally available.

All other values are used to identify version strings located in one of the other Symbol Version sections. The value itself is not the version associated with the symbol. The string identified by the value defines the version of the symbol.

8.2. Version Definitions

Symbol definitions are contained in the special section `.gnu.version_d` which has a section type of SHT_GNU_verdef. The number of entries in this section is contained in the DT_VERDEFNUM entry of the Dynamic Section. The `sh_link` member of the section header points to the section that contains the strings referenced by this section.

Figure 8-1. Version Definition Entries

```
typedef struct {  
    Elfxx_Half    vd_version;  
    Elfxx_Half    vd_flags;  
    Elfxx_Half    vd_ndx;  
    Elfxx_Half    vd_cnt;
```

```

32         Elfxx_Word    vd_hash;
33         Elfxx_Word    vd_aux;
34         Elfxx_Word    vd_next;
35     } Elfxx_Verdef;

36     vd_version
37         Version revision. This value is currently set to 1, and will be reset if the versioning implementation is
38         incompatibly altered.

39     vd_flags
40         Version information flag bitmask.

41     vd_ndx
42         Version index numeric value referencing the SHT_GNU_versym section.

43     vd_cnt
44         Number of associated verdaux array entries.

45     vd_hash
46         Version name hash value (ELF hash function).

47     vd_aux
48         Offset to a corresponding entry in the verdaux array, in bytes.

49     vd_next
50         Offset to the next verdef entry, in bytes.

```

51 **Figure 8-2. Version Definition Auxiliary Entries**

```

52     typedef struct {
53         Elfxx_Word    vda_name;
54         Elfxx_Word    vda_next;
55     } Elfxx_Verdaux;

56     vda_name
57         Offset to the version or dependency name string in the section header, in bytes.

58     vda_next
59         Offset to the next verdaux entry, in bytes.

```

8.3. Version Requirements

```

60     Symbol definitions are contained in the special section .gnu.version_r which has a section type of
61     SHT_GNU_verneed. The number of entries in this section is contained in the DT_VERNEEDNUM entry of the Dynamic
62     Section. The sh_link member of the section header points to the section that contains the strings referenced by this
63     section.

```

Figure 8-3. Version Needed Entries

```

typedef struct {
    Elfxx_Half    vn_version;
    Elfxx_Half    vn_cnt;
    Elfxx_Word    vn_file;
    Elfxx_Word    vn_aux;
    Elfxx_Word    vn_next;
} Elfxx_Verneed;

```

`vn_version`

Version of structure. This value is currently set to 1, and will be reset if the versioning implementation is incompatibly altered.

`vn_cnt`

Number of associated verneed array entries.

`vn_file`

Offset to the file name string in the section header, in bytes.

`vn_aux`

Offset to a corresponding entry in the vernaux array, in bytes.

`vn_next`

Offset to the next verneed entry, in bytes.

Figure 8-4. Version Needed Auxiliary Entries

```

typedef struct {
    Elfxx_Word    vna_hash;
    Elfxx_Half    vna_flags;
    Elfxx_Half    vna_other;
    Elfxx_Word    vna_name;
    Elfxx_Word    vna_next;
} Elfxx_Vernaux;

```

`vna_hash`

Dependency name hash value (ELF hash function).

`vna_flags`

Dependency information flag bitmask.

`vna_other`

Object file version identifier used in the `.gnu.version` symbol version array. Bit number 15 controls whether or not the object is hidden; if this bit is set, the object cannot be used and the static linker will ignore the symbol's presence in the object.

`vna_name`

Offset to the dependency name string in the section header, in bytes.

101 vna_next
 102 Offset to the next vernaux entry, in bytes.

8.4. Startup Sequence

103 When loading a sharable object, version definition data from the loaded object is analyzed to assure that it meets the
 104 version requirements of the calling object. The dynamic loader retrieves the entries in the caller's Elfxx_Verneed array
 105 and attempts to find matching definition information in the loaded Elfxx_Verdef table.

106 Each object and dependency is tested in turn. If a symbol definition is missing, the loader returns an error. A warning
 107 is issued instead of a hard error when the vna_flags bit for VER_FLG_WEAK is set in the Elfxx_Vernaux entry.

108 When the versions referenced by undefined symbols in the loaded object are found, version availability is certified.
 109 The test completes without error and the object is made available.

8.5. Symbol Resolution

110 When symbol versioning is used in an object, relocations extend the performance of definition testing beyond the
 111 simple match of symbol name strings: the version of the reference shall also equal the name of the definition. The
 112 same index that is used in the symbol table can be referenced in the SHT_GNU_versym section, and the value of this
 113 index is then used to acquire name data. The corresponding requirement string is retrieved from the Elfxx_Verneed
 114 array, and likewise, the corresponding definition string from the Elfxx_Verdef table.

115 Bit number 15 of the version symbol controls whether or not the object is hidden; if this bit is set, the object cannot be
 116 used and the static linker will ignore the symbol's presence in the object.

117 Results differ in the interaction of objects that variously use symbol versioning.

- 118 • The object with the reference and the object with the definitions may both use versioning. All described matching is
 119 processed in this case. A fatal error is triggered when no matching definition can be found in the object whose name
 120 is the one referenced by the vn_name element in the Elfxx_Verneed entry.
- 121 • The object with the reference may not use versioning, while the object with the definitions does. In this instance,
 122 only the definition with index numbers 1 and 2 will be used in the reference match, the same identified by the static
 123 linker as the base definition. In infrequent cases where the static linker was not used, as in calls to dlopen(), a
 124 version that does not have the base definition index is acceptable as long as it is the only version for which the
 125 symbol is defined.
- 126 • The object with the reference may use versioning, but the object with the definitions specifies none. A matching
 127 symbol is accepted in this case. A fatal error is triggered in the unlikely event that a corruption in the required
 128 symbols list obscured an outdated object file and caused a match on the object filename in the Elfxx_Verneed entry.
- 129 • Finally, both the object with the reference and the object with the definitions may not use versioning. The behavior
 130 in this instance defaults to pre-existing symbol rules.

Chapter 9. ABI note tag

- 1 Every executable shall contain a section named `.note.ABI-tag` of type `SHT_NOTE`. This section is structured as a
- 2 note section as documented in the ELF spec. The section shall contain at least the following entry. The `name` field
- 3 (`namesz/name`) contains the string "GNU". The `type` field shall be 1. The `descsz` field shall be at least 16, and the first
- 4 16 bytes of the `desc` field shall be as follows.
- 5 The first 32-bit word of the `desc` field shall be 0 (this signifies a Linux executable). The second, third, and fourth
- 6 32-bit words of the `desc` field contain the earliest compatible kernel version. For example, if the 3 words are 2, 2, and
- 7 5, this signifies a 2.2.5 kernel.

III. Dynamic Linking

Chapter 10. Program Loading and Dynamic Linking

- 1 LSB-conforming implementations shall support the object file information and system actions that create running
- 2 programs as specified in the System V ~~Application Binary Interface, Edition 4.1~~ABI and System V ~~Application~~
- 3 ~~Binary Interface DRAFT 17 December 2003~~ABI Update and as supplemented by this document and an
- 4 architecture-specific LSB specification.
- 5 Any shared object that is loaded shall contain sufficient DT_NEEDED records to satisfy the symbols on the shared
- 6 library.

Chapter 11. Program Header

In addition to the Segment Types defined in the System V Application Binary Interface, Edition 4.1 ABI and System V Application Binary Interface—DRAFT—17 December 2003 ABI Update the following Segment Types shall also be supported.

Table 11-1. Linux Segment Types

Name	Value
PT_GNU_EH_FRAME	0x6474e550
PT_GNU_STACK	0x6474e551

PT_GNU_EH_FRAME

The array element specifies the location and size of the exception handling information as defined by the .eh_frame_hdr section.

PT_GNU_STACK

The p_flags member specifies the permissions on the segment containing the stack and is used to indicate whether the stack should be executable. The absence of this header indicates that the stack will be executable.

Chapter 12. Dynamic Entries

A dynamic entry's *d_tag* member controls the interpretation of *d_un*.

12.1. Dynamic Entries

12.1.1. ELF Dynamic Entries

The following dynamic entries are defined in the System V Application Binary Interface, Edition 4.1ABI and System V Application Binary Interface — DRAFT — 17 December 2003ABI Update.

DT_BIND_NOW

Process relocations of object

DT_DEBUG

For debugging; unspecified

DT_FINI

Address of termination function

DT_HASH

Address of symbol hash table

DT_HIPROC

End of processor-specific

DT_INIT

Address of init function

DT_JMPREL

Address of PLT relocs

DT_LOPROC

Start of processor-specific

DT_NEEDED

Name of needed library

DT_NULL

Marks end of dynamic section

DT_PLTREL

Type of reloc in PLT

26	DT_PLTRELSZ
27	Size in bytes of PLT relocs
28	DT_REL
29	Address of Rel relocs
30	DT_RELA
31	Address of Rela relocs
32	DT_RELAENT
33	Size of one Rela reloc
34	DT_RELASZ
35	Total size of Rela relocs
36	DT_RELENT
37	Size of one Rel reloc
38	DT_RELSZ
39	Total size of Rel relocs
40	DT_RPATH
41	Library search path
42	DT_SONAME
43	Name of shared object
44	DT_STRSZ
45	Size of string table
46	DT_STRTAB
47	Address of string table
48	DT_SYMBOLIC
49	Start symbol search here
50	DT_SYMENT
51	Size of one symbol table entry
52	DT_SYMTAB
53	Address of symbol table
54	DT_TEXTREL
55	Reloc might modify .text

12.1.2. Additional Dynamic Entries

56 The following dynamic entries are defined here.

57 **DT_ADDRRNGHI**

58 Values from DT_ADDRRNGLO through DT_ADDRRNGHI are reserved for definition by an archLSB.

59 **DT_ADDRRNGLO**

60 Values from DT_ADDRRNGLO through DT_ADDRRNGHI are reserved for definition by an archLSB.

61 **DT_AUXILIARY**

62 Shared object to load before self

63 **DT_FILTER**

64 Shared object to get values from

65 **DT_FINI_ARRAY**

66 The address of an array of pointers to termination functions.

67 **DT_FINI_ARRAYSZ**

68 Size in bytes of DT_FINI_ARRAY

69 **DT_HIOS**

70 Values from DT_LOOS through DT_HIOS are reserved for definition by specific operating systems.

71 **DT_INIT_ARRAY**

72 The address of an array of pointers to initialization functions.

73 **DT_INIT_ARRAYSZ**

74 Size in bytes of DT_INIT_ARRAY

75 **DT_LOOS**

76 Values from DT_LOOS through DT_HIOS are reserved for definition by specific operating systems.

77 **DT_NUM**

78 Number of dynamic entry tags defined (excepting reserved ranges).

79 **DT_POSFLAG_1**

80 Flags for DT_* entries, effecting the following DT_* entry

81 **DT_RELCOUNT**

82 All Elf32_Rel R_*_RELATIVE relocations have been placed into a single block and this entry specifies the

83 number of entries in that block. This permits ld.so.1 to streamline the processing of RELATIVE relocations.

84	DT_SYMINENT
85	Entry size of syminfo
86	DT_SYMINFO
87	Address of the Syminfo table.
88	DT_SYMINSZ
89	Size of syminfo table (in bytes)
90	DT_VALRNGHI
91	Entries which fall between DT_VALRNGHI & DT_VALRNGLO use the Dyn.d_un.d_val field of the Elf*_Dyn
92	structure.
93	DT_VALRNGLO
94	Entries which fall between DT_VALRNGHI & DT_VALRNGLO use the Dyn.d_un.d_val field of the Elf*_Dyn
95	structure.
96	DT_VERDEF
97	Address of version definition table
98	DT_VERDEFNUM
99	Number of version definitions
100	DT_VERNEED
101	Address of table with needed versions
102	DT_VERNEEDNUM
103	Number of needed versions
104	DT_VERSYM
105	Address of the table provided by the .gnu.version section.

Linux Standard Base Specification

2

3 **Linux Standard Base Specification**

Table of Contents

I. Base Libraries	45
1. Libraries	1
1.1. Program Interpreter.....	1
1.2. Interfaces for libc	1
1.2.1. RPC	1
1.2.1.1. Interfaces for RPC	1
1.2.2. System Calls	2
1.2.2.1. Interfaces for System Calls	2
1.2.3. Standard I/O	4
1.2.3.1. Interfaces for Standard I/O	4
1.2.4. Signal Handling.....	5
1.2.4.1. Interfaces for Signal Handling	5
1.2.5. Localization Functions	6
1.2.5.1. Interfaces for Localization Functions	6
1.2.6. Socket Interface.....	7
1.2.6.1. Interfaces for Socket Interface	7
1.2.7. Wide Characters	8
1.2.7.1. Interfaces for Wide Characters	8
1.2.8. String Functions	9
1.2.8.1. Interfaces for String Functions.....	9
1.2.9. IPC Functions.....	10
1.2.9.1. Interfaces for IPC Functions	10
1.2.10. Regular Expressions	11
1.2.10.1. Interfaces for Regular Expressions	11
1.2.11. Character Type Functions	12
1.2.11.1. Interfaces for Character Type Functions.....	12
1.2.12. Time Manipulation.....	12
1.2.12.1. Interfaces for Time Manipulation	12
1.2.13. Terminal Interface Functions	13
1.2.13.1. Interfaces for Terminal Interface Functions.....	13
1.2.14. System Database Interface	14
1.2.14.1. Interfaces for System Database Interface.....	14
1.2.15. Language Support	15
1.2.15.1. Interfaces for Language Support.....	15
1.2.16. Large File Support.....	15
1.2.16.1. Interfaces for Large File Support	15
1.2.17. Standard Library.....	16
1.2.17.1. Interfaces for Standard Library	16
1.3. Data Definitions for libc	18
1.3.1. assert.h.....	18
1.3.2. ctype.h	18
1.3.3. dirent.h	18
1.3.4. errno.h	19

1.3.5. fcntl.h	21
1.3.6. fmtmsg.h	22
1.3.7. fnmatch.h	23
1.3.8. ftw.h	23
1.3.9. getopt.h	23
1.3.10. glob.h	24
1.3.11. grp.h	25
1.3.12. iconv.h	25
1.3.13. inttypes.h	25
1.3.14. langinfo.h	25
1.3.15. limits.h	27
1.3.16. locale.h	27
1.3.17. net/if.h	28
1.3.18. netdb.h	30
1.3.19. netinet/in.h	31
1.3.20. netinet/tcp.h	33
1.3.21. netinet/udp.h	33
1.3.22. nl_types.h	33
1.3.23. pty.h	33
1.3.24. pwd.h	33
1.3.25. regex.h	34
1.3.26. rpc/auth.h	35
1.3.27. rpc/clnt.h	36
1.3.28. rpc/rpc_msg.h	38
1.3.29. rpc/svc.h	40
1.3.30. rpc/types.h	41
1.3.31. rpc/xdr.h	41
1.3.32. sched.h	42
1.3.33. search.h	42
1.3.34. setjmp.h	42
1.3.35. signal.h	43
1.3.36. stddef.h	47
1.3.37. stdio.h	47
1.3.38. stdlib.h	48
1.3.39. sys/file.h	49
1.3.40. sys/ipc.h	49
1.3.41. sys/mman.h	49
1.3.42. sys/msg.h	50
1.3.43. sys/param.h	50
1.3.44. sys/poll.h	50
1.3.45. sys/resource.h	50
1.3.46. sys/sem.h	51
1.3.47. sys/shm.h	52
1.3.48. sys/socket.h	52
1.3.49. sys/stat.h	54
1.3.50. sys/time.h	55
1.3.51. sys/timeb.h	55
1.3.52. sys/times.h	56

1.3.53. sys/types.h	56
1.3.54. sys/un.h	57
1.3.55. sys/utsname.h	57
1.3.56. sys/wait.h	58
1.3.57. syslog.h	58
1.3.58. termios.h	59
1.3.59. time.h	61
1.3.60. ulimit.h	61
1.3.61. unistd.h	61
1.3.62. utime.h	65
1.3.63. utmp.h	65
1.3.64. wchar.h	66
1.3.65. wctype.h	66
1.3.66. wordexp.h	66
1.4. Interface Definitions for libc	67
_IO_feof	67
_IO_getc	67
_IO_putc	68
_IO_puts	68
__assert_fail	69
__ctype_b_loc	69
__ctype_get_mb_cur_max	70
__ctype_tolower_loc	70
__ctype_toupper_loc	71
__cxa_atexit	71
__daylight	72
__environ	72
__errno_location	73
__fpending	73
__getpagesize	73
__getpgid	74
__h_errno_location	74
__isinf	75
__isinf	75
__isinfl	76
__isnan	76
__isnanf	77
__isnanl	77
__libc_current_sigrtmax	78
__libc_current_sigrtmin	78
__libc_start_main	79
__lxstat	79
__mempcpy	80
__rawmemchr	80
__register_atfork	81
__sigsetjmp	81
__stpcpy	82
__strdup	82

__strtod_internal	83
__strtof_internal	83
__strtok_r	84
__strtol_internal	84
__strtold_internal	85
__strtoll_internal	85
__strtoul_internal	86
__strtoull_internal	86
__sysconf	87
__sysv_signal	87
__timezone	88
__tzname	88
__wcstod_internal	88
__wcstof_internal	89
__wcstol_internal	89
__wcstold_internal	90
__wcstoul_internal	90
__xmknod	91
__xstat	91
__xstat64	92
_environ	93
_nl_msg_cat_cntr	94
_obstack_begin	94
_obstack_newchunk	95
_sys_errlist	95
_sys_siglist	95
acct	97
adjtime	98
adjtimex	99
asprintf	100
bind_textdomain_codeset	101
bindresvport	103
bindtextdomain	104
cfmakeraw	105
cfsetspeed	106
creat	107
daemon	108
dcgettext	108
dcngettext	110
dgettext	113
dngettext	114
err	115
error	116
errx	117
fcntl	118
fflush_unlocked	118
fgetwc_unlocked	118
flock	119

fopen	120
freopen	120
getdomainname	121
gethostbyname_r	122
getloadavg	123
getopt	123
getopt_long	126
getopt_long_only	127
gettext	129
gettutent	130
gettutent_r	131
glob64	132
globfree64	134
initgroups	134
ioctl	136
sockio	137
iswtypekill	139
killmbsnrtowcs	141
mbsnrtowesmemmem	142
memmemmemrchr	144
memrchrnggettext	145
nggettextobstack_free	146
obstack_freeopen	146
openopterr	147
opterrtoptind	148
optindtoptopt	148
optoptpmap_getport	149
pmap_getportset	150
pmap_unset	150
pmap_unsetpsignal	151
psignalrandom_r	151
random_rsetbuffer	152
setbufferdomainname	152
setdomainnamesetgroups	154
setgroupssethostid	155
sethostidsethostname	155
sethostnameetsockopt	157
setsockoptsetutent	158
setutentsetsigandset	159
sigandsetsigblock	159
sigblocksiggetmask	160
siggetmasksigisemptyset	161
sigisemptysetsigorset	161
sigorsetsigreturn	163
sigreturnstime	163
stimestpcpy	165
stpepystpncpy	166
stpncpystrcasestr	167

streasestrerror_r.....	168
strerror_rstrfry.....	168
strfrystrndup.....	169
strndupstrnlen.....	169
strlenstrptime.....	170
strptimestrsep.....	171
strsepstrsignal.....	171
strsignalstrtoq.....	173
strtok_rstrtouq.....	174
strtoqstrverscmp.....	176
strtouqsvc_register.....	177
strversempsvc_run.....	177
svc_registersendreply.....	178
svc_runtcp_create.....	178
sve_sendreplysvcudp_create.....	179
svetcp_createsystem.....	180
svcudp_creategettextdomain.....	181
systemunlink.....	182
gettextdomainvasprintf.....	182
unlinkvdprintf.....	183
vasprintfverrx.....	184
vdprintfvsyslog.....	184
verrxwait3.....	185
vsyslogwait4.....	185
wait3waitpid.....	187
wait4warn.....	188
waitpidwarnx.....	189
warnwcpcpy.....	190
warnxwcpncpy.....	190
wcpcpywcscasecmp.....	191
wcpncpywcscdup.....	191
wcscasecmpwcscncasecmp.....	192
wcscdupwcscnlen.....	193
wcscncasecmpwcscnrtombs.....	194
wcscnlenwcstouq.....	195
wcscnrtombswcstouq.....	196
wcstouqxd_r_u_int.....	196
wcstouq1.5. Interfaces for libm.....	196
xd_r_u_int1.5.1. Math.....	197
1.5.1.1. Interfaces for libmMath.....	197
1.5.1. Math1.6. Data Definitions for libm.....	200
1.5.1.1. Interfaces for Math1.6.1. complex.h.....	200
1.6. Data Definitions for libm1.6.2. math.h.....	200
1.6.1. complex.h1.7. Interfaces for libpthread.....	201
1.6.2. math.h1.7.1. Realtime Threads.....	201
1.7.1.1. Interfaces for libpthreadRealtime Threads.....	201
1.7.1.2. Advanced Realtime Threads.....	201
1.7.1.2.1. Interfaces for Advanced Realtime Threads.....	201

1.7.2. Advanced Realtime3. Posix Threads.....	202
1.7.23.1. Interfaces for Advanced RealtimePosix Threads.....	202
1.7.3. Posix Threads1.8. Data Definitions for libpthread.....	203
1.7.38.1. Interfaces for Posix Threadspthread.h.....	203
1.8. Data Definitions for libpthread1.8.2. semaphore.h	206
1.8.1. pthread.h1.9. Interface Definitions for libpthread.....	206
1.8.2. semaphore.h pthread_cleanup_pop	207
1.9. Interface Definitions for libpthread pthread_cleanup_push	207
pthread_cleanup_pop1.10. Interfaces for libgcc_s	207
pthread_cleanup_push1.10.1. Unwind Library	207
1.10.1.1. Interfaces for libgcc_sUnwind Library	207
1.10.1. Unwind Library1.11. Data Definitions for libgcc_s	207
1.10.1.1. Interfaces for Unwind Library1.11.1. unwind.h.....	208
1.11. Data Definitions for libgcc_sl1.12. Interfaces for libdl	208
1.112.1. unwind.hDynamic Loader	209
1.12.1.1. Interfaces for libdlDynamic Loader	209
1.12.1. Dynamic Loader1.13. Data Definitions for libdl	209
1.12.1.1. Interfaces for Dynamic Loader1.13.1. dlfcn.h.....	209
1.13. Data14. Interface Definitions for libdl.....	210
1.13.1. dlfcn.hdladdr	210
1.14. Interface Definitions for libdlldlopen	212
dladdrldlsym.....	213
ldlopen1.15. Interfaces for libcrypt.....	213
ldsym1.15.1. Encryption	213
1.15.1.1. Interfaces for libcryptEncryption.....	213
1.15.1. Encryption1.16. Interfaces for libpam	213
1.15.1.1. Interfaces for Encryption1.16.1. Pluggable Authentication API.....	214
1.16.1.1. Interfaces for libpamPluggable Authentication API.....	214
1.16.1. Pluggable Authentication API1.17. Data Definitions for libpam	214
1.16.1.1. Interfaces for Pluggable Authentication API1.17.1. security/pam_appl.h	214
1.17. Data18. Interface Definitions for libpam	216
1.17.1. security/pam_appl.hpam_acct_mgmt.....	217
1.18. Interface Definitions for libpampam_authenticate	218
pam_acct_mgmtpam_chauthtok.....	220
pam_authenticatepam_close_session	221
pam_chauthtokend	222
pam_close_sessionpam_fail_delay	223
pam_endget_item	224
pam_fail_delaypam_getenvlist.....	225
pam_get_itemopen_session.....	226
pam_getenvlistset_item	227
pam_open_sessionsetcred	229
pam_set_itemstart	230
pam_setcredstrerror.....	231
pam_startII. Utility Libraries.....	233
pam_strerror2. utility Libraries	234
II. Utility Libraries2.1. Interfaces for libz	234

gettextgroupadd	271
grepgroupdel	271
groupaddgroupmod	272
groupdelgroups	273
groupmodgunzip	273
groupsgzip	274
gunziphostname	276
gzipinstall	279
hostnameinstall_initd	280
installipcrm	281
install_initdipcs	282
ipermkillall	283
ipeslpr	284
killalls	286
lprlsb_release	287
lsmd4	288
lsb_releasemd5sum	289
m4mknod	290
md5summktemp	291
mknodmore	292
mktempmount	295
moremsgfmt	298
mountnewgrp	305
msgfmtod	306
newgrppasswd	308
odpatch	309
passwdpidof	309
patchremove_initd	310
pidofrenice	310
remove_initdsed	311
renicesendmail	312
sedshutdown	315
sendmailsu	317
shutdownsync	318
sutar	318
syneumount	319
taruseradd	320
umountuserdel	323
useraddusermod	323
userdelxargs	325
usermodIV. Execution Environment.....	327
xargs 4. File System Hierarchy	328
IV. Execution Environment 4.1. /dev	328
4. File System Hierarchy 5. Additional Recommendations	329
45.1. /dev Minimal granted Directory and File permissions	329
5. Additional 2. Recommendations for applications on ownership and permissions	329
5.2.1. Minimal granted Directory and File permissions Write Permissions	329

	5.2. Recommendations for applications on ownership and permissions	329
	5.2.1. Directory Write	329
	5.2.2. File Write	329
	5.2.3. File Read and execute Permissions	330
	5.2.4. Suid and Sgid Permissions	330
	5.2.5. Privileged users	330
	5.2.6. Changing permissions	330
	5.2.7. Removable Media (Cdrom, Floppy, etc.)	331
	5.2.8. Installable applications	331
	6. Additional Behaviors	333
	6.1. Mandatory Optional Behaviors	333
	6.1.1. Special Requirements	333
	7. Localization	333
	7.1. Regular Expressions	333
	7.2. Pattern Matching Notation	333
	V. System Initialization	334
	7.2. Filename Globbing	335
	8. System Initialization	335
	8.1. Cron Jobs	336
	8.2. Init Script Actions	337
	8.3. Comment Conventions for Init Scripts	338
	8.4. Installation and Removal of init.d Files	339
	8.5. Run Levels	339
	8.6. Facility Names	340
	8.7. Script Names	340
	8.8. Init Script Functions	340
	VI. Users & Groups	343
	9. Users & Groups	344
	9.1. User and Group Database	344
	9.2. User & Group Names	344
	9.3. UID Ranges	345
	9.4. Rationale	345
	A. Alphabetical Listing of Interfaces	346
	A.1. libc	346
	A.2. libcrypt	363
	A.3. libdl	367
	A.4. libm	367
	A.5. libncurses	377
	A.6. libpam	381
	A.7. libpthread	382
	A.8. libutil	384
	A.9. libz	384
1	A.10. libz	000
2	A.11. libz	000
3	A.12. libz	000
4	A.13. libz	000
5	A.14. libz	000
6	A.15. libz	000

7	A.14. libpthread.....	.000
8	A.15. libpam.....	.000

List of Tables

1-1. libc Definition.....	1
1-2. libc - RPC Function Interfaces	1
1-3. libc - System Calls Function Interfaces	2
1-4. libc - Standard I/O Function Interfaces	4
1-5. libc - Standard I/O Data Interfaces	5
1-6. libc - Signal Handling Function Interfaces	5
1-7. libc - Signal Handling Data Interfaces.....	6
1-8. libc - Localization Functions Function Interfaces	6
1-9. libc - Localization Functions Data Interfaces	7
1-10. libc - Socket Interface Function Interfaces	7
1-11. libc - Socket Interface Deprecated Function Interfaces	8
1-12. libc - Wide Characters Function Interfaces	8
1-13. libc - String Functions Function Interfaces.....	9
1-14. libc - IPC Functions Function Interfaces	10
1-15. libc - Regular Expressions Function Interfaces	11
1-16. libc - Regular Expressions Deprecated Function Interfaces	11
1-17. libc - Regular Expressions Deprecated Data Interfaces.....	11
1-18. libc - Character Type Functions Function Interfaces.....	12
1-19. libc - Time Manipulation Function Interfaces	12
1-20. libc - Time Manipulation Deprecated Function Interfaces	13
1-21. libc - Time Manipulation Data Interfaces.....	13
1-22. libc - Terminal Interface Functions Function Interfaces.....	14
1-23. libc - System Database Interface Function Interfaces.....	14
1-24. libc - Language Support Function Interfaces.....	15
1-25. libc - Large File Support Function Interfaces	15
1-26. libc - Standard Library Function Interfaces.....	16
1-27. libc - Standard Library Data Interfaces	17
1-28. libm Definition 1-1. Examples	176
1-29. libm - Math Function Interfaces Definition.....	197
1-30. libm - Math Data Function Interfaces.....	197
1-31. libpthread Definition 1-31. libm - Math Data Interfaces	200
1-32. libpthread - Posix Threads Function Interfaces Definition.....	201
1-33. libgcc_s Definition 1-33. libpthread - Posix Threads Function Interfaces	202
1-34. libdl libgcc_s Definition	207
1-35. libdl - Dynamic Loader Function Interfaces Definition	208
1-36. libcrypt Definition 1-36. libdl - Dynamic Loader Function Interfaces	209
1-37. libcrypt - Encryption Function Interfaces Definition.....	213
1-38. libpam Definition 1-38. libcrypt - Encryption Function Interfaces	213
1-39. libpam - Pluggable Authentication API Function Interfaces Definition.....	214
2-1. libz Definition 1-40. libpam - Pluggable Authentication API Function Interfaces	214
2-21. libz - Compression Library Function Interfaces Definition.....	234
2-3. libncurses Definition 2-2. libz - Compression Library Function Interfaces	234
2-43. libncurses - Curses Function Interfaces Definition.....	237

	2-54. libncurses - Curses DataFunction Interfaces	237
	2-6. libutil Definition2-5. libncurses - Curses Data Interfaces	240
	2-76. libutil - Utility Functions Function InterfacesDefinition	245
	3-1. Commands and Utilities2-7. libutil - Utility Functions Function Interfaces	246
	9-1. Required User & Group Names3-1. Commands and Utilities.....	254
	9-2. Optional User & Group Names3-1. Escape Sequences	301
	A-1. libXt11 Function Interfaces9-1. Required User & Group Names.....	344
	A-2. libXt11 Data Interfaces9-2. Optional User & Group Names	345
	A-3. libXt1. libc Function Interfaces	354
	A-4. libXt2. libc Data Interfaces.....	363
	A-5. libm3. libcrypt Function Interfaces.....	363
	A-6. libm Data4. libdl Function Interfaces	367
	A-7. libGL5. libm Function Interfaces	367
	A-8. libXext Function6. libm Data Interfaces.....	372
	A-9. libICE7. libncurses Function Interfaces.....	378
	A-10. libSM Function8. libncurses Data Interfaces.....	381
	A-11. libdl9. libpam Function Interfaces	381
	A-12. libcrypt10. libpthread Function Interfaces	382
	A-13. libz11. libutil Function Interfaces.....	384
	A-14. libncurses12. libz Function Interfaces	384
1	A-15. libncurses Data Interfaces.....	000
2	A-16. libutil Function Interfaces.....	000
3	A-17. libc Function Interfaces	000
4	A-18. libc Data Interfaces.....	000
5	A-19. libpthread Function Interfaces	000
6	A-20. libpam Function Interfaces	000

I. Base Libraries

Chapter 1. Libraries

An LSB-conforming implementation shall support some base libraries which provide interfaces for accessing the operating system, processor and other hardware in the system.

1.1. Program Interpreter

The Program Interpreter is specified in the appropriate architecture-specific LSB specification.

1.2. Interfaces for libc

Table 1-1 defines the library name and shared object name for the libc library

Table 1-1. libc Definition

Library:	libc
SONAME:	See archLSB.

The behavior of the interfaces in this library is specified by the following specifications:

Large File Support

Linux Standard Base this specification

CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606) SUSv2

ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3) System V Interface Definition, SVID Issue 3 (ISBN 0201566524)

System V Interface Definition, Fourth Edition SVID Issue 4

1.2.1. RPC

1.2.1.1. Interfaces for RPC

An LSB conforming implementation shall provide the generic functions for RPC specified in Table 1-2, with the full functionality as described in the referenced underlying specification.

Table 1-2. libc - RPC Function Interfaces

authnone_create authnone_create [1]	pmap_unset pmap_unset [2]	svcerr_weakauth svcerr_weakauth [3]	xdr_float xdr_float [3]	xdr_u_char xdr_u_char [3]
clnt_create clnt_create [1]	setdomainname setdomainname [2]	svctcp_create svctcp_create [2]	xdr_free xdr_free [3]	xdr_u_int xdr_u_int [2]
clnt_percreateerror clnt_percreateerror [1]	svc_getreqset svc_getreqset [3]	svcudp_create svcudp_create [2]	xdr_int xdr_int [3]	xdr_u_long xdr_u_long [3]
clnt_permnoclnt_perr	svc_registersvc_regi	xdr_accepted_reply	xdr_long xdr_long	xdr_u_short xdr_u_s

no [1]	ster [2]	xdr_accepted_reply [3]	[3]	hort [3]
clnt_perrorclnt_perror [1]	sve_runsvcsvc_run [2]	xdr_arrayxdr_array [3]	xdr_opaque_xdr_opaque [3]	xdr_unionxdr_union [3]
clnt_spcreateerrorclnt_spcreateerror [1]	sve_sendreplysvc_sendreply [2]	xdr_boolxdr_bool [3]	xdr_opaque_authxdr_opaque_auth [3]	xdr_vectorxdr_vector [3]
clnt_sperrnoclnt_sperrno [1]	sve_err_authsvc_err_auth [3]	xdr_bytesxdr_bytes [3]	xdr_pointerxdr_pointer [3]	xdr_voidxdr_void [3]
clnt_sperrorclnt_sperror [1]	sve_err_decodesvc_err_decode [3]	xdr_callhdrxdr_callhdr [3]	xdr_reference_xdr_reference [3]	xdr_wrapstringxdr_wrapstring [3]
getdomainnamegetdomainname [2]	sve_err_noprocsvc_err_noproc [3]	xdr_callmsgxdr_callmsg [3]	xdr_rejected_replyxdr_rejected_reply [3]	xdrmem_createxdrmem_create [3]
key_decryptsessionkey_decryptsession [3]	sve_err_noprogsvc_err_noprogram [3]	xdr_charxdr_char [3]	xdr_replymsgxdr_replymsg [3]	xdrrec_createxdrrec_create [3]
pmap_getportpmap_getport [2]	sve_err_progverssvc_err_progvers [3]	xdr_doublexdr_double [3]	xdr_shortxdr_short [3]	xdrrec_eofxdrrec_eof [3]
pmap_setpmap_set [2]	sve_err_systemerrsvc_err_systemerr [3]	xdr_enumxdr_enum [3]	xdr_stringxdr_string [3]	

Referenced Specification(s)

- [1]. System V Interface Definition, Fourth Edition, SVID Issue 4
- [2]. Linux Standard Base, this specification
- [3]. System V Interface Definition, SVID Issue 3 (ISBN 0201566524)

1.2.2. System Calls

1.2.2.1. Interfaces for System Calls

An LSB conforming implementation shall provide the generic functions for System Calls specified in Table 1-3, with the full functionality as described in the referenced underlying specification.

Table 1-3. libc - System Calls Function Interfaces

__fxstat__fxstat [1]	fchmodfchmod [2]	getwdgetwd [2]	readread [2]	setlimitsetlimit [2]
__getpgid__getpgid [1]	fchownfchown [2]	initgroupsinitgroups [1]	readdirreaddir [2]	setlimit64setlimit64 [3]
__lxstat__lxstat [1]	fcntlfcntl [1]	ioctlioctl [1]	readdir_rreaddir_r [2]	setsidsetsid [2]
__xmknod__xmknod	fdatasyncfdatasync	killkill [1]	readlinkreadlink [2]	setuidsetuid [2]

d [1]	[2]			
__xstat__xstat [1]	flockflock [1]	killpgkillpg [2]	readvreadv [2]	sleepsleep [2]
accessaccess [2]	forkfork [2]	lchownlchown [2]	renamereaname [2]	statvfsstatvfs [2]
acctacct [1]	fstatvfsfstatvfs [2]	linklink [2]	rmdirmdir [2]	time [1]
alarmalarm [2]	fsyncfsync [2]	lockflockf [2]	sbrksbrk [4]	symlink [2]
brkbrk [4]	ftime [2]	lseekseek [2]	sched_get_priority_maxsched_get_priority_max [2]	synesync [2]
chdirchdir [2]	ftruncateftruncate [2]	mkdirmkdir [2]	sched_get_priority_minsched_get_priority_min [2]	sysconfsysconf [2]
chmodchmod [2]	getcontextgetcontext [2]	mknfifomknfif [2]	sched_getparamsched_getparam [2]	time [2]
chownchown [2]	getegidgetegid [2]	mlockmlock [2]	sched_getschedulersched_getscheduler [2]	times [2]
chrootchroot [4]	geteuidgeteuid [2]	mlockallmlockall [2]	sched_rr_get_intervalsched_rr_get_interval [2]	truncatetruncate [2]
clockclock [2]	getgidgetgid [2]	mmapmmap [2]	sched_setparamsched_setparam [2]	ulimit [2]
closeclose [2]	getgroupsgetgroups [2]	mprotectmprotect [2]	sched_setschedulersched_setscheduler [2]	umaskumask [2]
closedirclosedir [2]	getitimergetitimer [2]	msyncmsync [2]	sched_yieldsched_yield [2]	uname [2]
creatcreat [1]	getloadavggetloadavg [1]	munlockmunlock [2]	selectselect [2]	unlink [1]
dupdup [2]	getpagesizegetpagesize [4]	munlockallmunlockall [2]	setcontextsetcontext [2]	utime [2]
dup2dup2 [2]	getpgidgetpgid [2]	munmapmunmap [2]	setgidsetgid [2]	utimes [2]
execlexecl [2]	getpgrpgetpgrp [2]	nanosleepnanosleep [2]	setuidsetuid [2]	vforkvfork [2]
execleexecle [2]	getpidgetpid [2]	nice [2]	setgidsetgid [2]	wait [2]
execlpexeclp [2]	getppidgetppid [2]	openopen [1]	setitimer [2]	wait3wait3 [1]

execv execv [2]	getpriority getpriority [2]	opendir opendir [2]	setpgid setpgid [2]	wait4 wait4 [1]
execve execve [2]	getrlimit getrlimit [2]	pathconf pathconf [2]	setpgrp setpgrp [2]	waitpid waitpid [1]
execvp execvp [2]	getrusage getrusage [2]	pause pause [2]	setpriority setpriority [2]	write write [2]
exit exit [2]	getsid getsid [2]	pipe pipe [2]	setregid setregid [2]	writex writex [2]
fchdir fchdir [2]	getuid getuid [2]	poll poll [2]	setreuid setreuid [2]	

Referenced Specification(s)

[1]. Linux Standard Basethis specification

[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

[3]. Large File Support

[4]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0, C606) SUSv2

1.2.3. Standard I/O

1.2.3.1. Interfaces for Standard I/O

An LSB conforming implementation shall provide the generic functions for Standard I/O specified in Table 1-4, with the full functionality as described in the referenced underlying specification.

Table 1-4. libc - Standard I/O Function Interfaces

_IO_feof _IO_feof [1]	fgetpos fgetpos [2]	fsetpos fsetpos [2]	putchar putchar [2]	sscanf sscanf [2]
_IO_getc _IO_getc [1]	fgets fgets [2]	ftell ftell [2]	putchar_unlocked putchar_unlocked [2]	telldir telldir [2]
_IO_pute _IO_putc [1]	fgetwc_unlocked fgetwc_unlocked [1]	felloftello felloftello [2]	puts puts [2]	tempnam tempnam [2]
_IO_puts _IO_puts [1]	fileno fileno [2]	fwrite fwrite [2]	putw putw [3]	ungetc ungetc [2]
asprintf asprintf [1]	flockfile flockfile [2]	gete getc [2]	remove remove [2]	vasprintf vasprintf [1]
clearerr clearerr [2]	fopen fopen [1]	getc_unlocked getc_unlocked [2]	rewind rewind [2]	vdprintf vdprintf [1]
ctermid ctermid [2]	fprintf fprintf [2]	getchar getchar [2]	rewinddir rewinddir [2]	vfprintf vfprintf [2]

fclose fclose [2]	fputc fputc [2]	getchar_unlocked tchar_unlocked [2]	scanf scanf [2]	vprintf vprintf [2]
fopen fopen [2]	fputs fputs [2]	getw getw [3]	seekdir seekdir [2]	vsnprintf vsnprintf [2]
ferror ferror [2]	fread fread [2]	perror perror [2]	setbuf setbuf [2]	vsprintf vsprintf [2]
fflush fflush [2]	fscanf fscanf [2]	printf printf [2]	setbuffer setbuffer [1]	
fflush_unlocked fflush_unlocked [1]	fseek fseek [2]	putc putc [2]	snprintf snprintf [2]	
fgetc fgetc [2]	fseeko fseeko [2]	putc_unlocked putc_unlocked [2]	sprintf sprintf [2]	

Referenced Specification(s)

[1]. Linux Standard Base this specification

[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

[3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0, C606) SUSv2

An LSB conforming implementation shall provide the generic data interfaces for Standard I/O specified in Table 1-5, with the full functionality as described in the referenced underlying specification.

Table 1-5. libc - Standard I/O Data Interfaces

stderr stderr [1]	stdin stdin [1]	stdout stdout [1]		
------------------------------	----------------------------	------------------------------	--	--

Referenced Specification(s)

[1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.2.4. Signal Handling

1.2.4.1. Interfaces for Signal Handling

An LSB conforming implementation shall provide the generic functions for Signal Handling specified in Table 1-6, with the full functionality as described in the referenced underlying specification.

Table 1-6. libc - Signal Handling Function Interfaces

__libc_current_sigrt max __libc_current_ sigrtmax [1]	sigaddset sigaddset [2]	sighold sighold [2]	sigpauses sigpause [2]	sigsuspend sigsuspend [2]
__libc_current_sigrt	sigaltstack sigaltstack	sigignore sigignore	sigpending sigpending	sigtimedwait sigtimedwait

min _libc_current_sigrtmin [1]	k [2]	[2]	ng [2]	dwait [2]
__sigsetjmp _sigsetjmp [1]	sigandset sigandset [1]	siginterrupt siginterrupt [2]	sigprocmask sigprocmask [2]	sigwait sigwait [2]
__sysv_signal _sysv_signal [1]	sigblock sigblock [1]	sigisemptyset sigisemptyset [1]	sigqueue sigqueue [2]	sigwaitinfo sigwaitinfo [2]
bsd_signal bsd_signal [2]	sigdelset sigdelset [2]	sigismember sigismember [2]	sigelse sigelse [2]	
psignal psignal [1]	sigemptyset sigemptyset [2]	siglongjmp siglongjmp [2]	sigreturn sigreturn [1]	
raiseraise [2]	sigfillset sigfillset [2]	signal signal [2]	sigset sigset [2]	
sigaction sigaction [2]	siggetmask siggetmask [1]	sigorset sigorset [1]	sigstack sigstack [3]	

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

[2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

[3]. ~~CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)~~SUSv2

An LSB conforming implementation shall provide the generic data interfaces for Signal Handling specified in Table 1-7, with the full functionality as described in the referenced underlying specification.

Table 1-7. libc - Signal Handling Data Interfaces

_sys_siglist _sys_siglist [1]				
--	--	--	--	--

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

1.2.5. Localization Functions

1.2.5.1. Interfaces for Localization Functions

An LSB conforming implementation shall provide the generic functions for Localization Functions specified in Table 1-8, with the full functionality as described in the referenced underlying specification.

Table 1-8. libc - Localization Functions Function Interfaces

bind_textdomain_codeset bind_textdomain_codeset [1]	catopen catopen [2]	dngettext dngettext [1]	iconv_open iconv_open [2]	setlocale setlocale [2]
--	--------------------------------	------------------------------------	--------------------------------------	------------------------------------

bindtextdomain bindtextdomain [1]	dcgettext dcgettext [1]	gettext gettext [1]	localeconv localeconv [2]	textdomain textdomain [1]
catclose catclose [2]	dcngettext dcngettext [1]	iconv iconv [2]	ngettext ngettext [1]	
catgets catgets [2]	dcgettext dcgettext [1]	iconv_close iconv_close [2]	nl_langinfo nl_langinfo [2]	

Referenced Specification(s)

[1]. Linux Standard Basethis specification

[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

An LSB conforming implementation shall provide the generic data interfaces for Localization Functions specified in Table 1-9, with the full functionality as described in the referenced underlying specification.

Table 1-9. libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr _nl_msg_cat_cntr [1]				
--	--	--	--	--

Referenced Specification(s)

[1]. Linux Standard Basethis specification

1.2.6. Socket Interface

1.2.6.1. Interfaces for Socket Interface

An LSB conforming implementation shall provide the generic functions for Socket Interface specified in Table 1-10, with the full functionality as described in the referenced underlying specification.

Table 1-10. libc - Socket Interface Function Interfaces

_h_errno_location _h_errno_location [1]	gethostid gethostid [2]	listen listen [2]	sendmsg sendmsg [2]	socketpair socketpair [2]
accept accept [2]	gethostname gethostname [2]	recv recv [2]	sendto sendto [2]	
bind bind [2]	getpeername getpeername [2]	recvfrom recvfrom [2]	setsockopt setsockopt [1]	
bindresvport bindresvport [1]	getsockname getsockname [2]	recvmsg recvmsg [2]	shutdown shutdown [2]	
connect connect [2]	getsockopt getsockopt [2]	send send [2]	socket socket [2]	

Referenced Specification(s)

[1]. Linux Standard Base this specification

[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

An LSB conforming implementation shall provide the generic deprecated functions for Socket Interface specified in Table 1-11, with the full functionality as described in the referenced underlying specification.

These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn in future releases of this specification.

Table 1-11. libc - Socket Interface Deprecated Function Interfaces

gethostname_r hostname_r [1]				
--	--	--	--	--

Referenced Specification(s)

[1]. Linux Standard Base this specification

1.2.7. Wide Characters

1.2.7.1. Interfaces for Wide Characters

An LSB conforming implementation shall provide the generic functions for Wide Characters specified in Table 1-12, with the full functionality as described in the referenced underlying specification.

Table 1-12. libc - Wide Characters Function Interfaces

__westod_internal_ _westod_internal [1]	mbstowcs mbstowcs [2]	wcsnlen wcsnlen [1]	wcstombs wcstombs [2]	wcstombs wcstombs [2]
__westof_internal_ westof_internal [1]	mbstowcs mbstowcs [1]	wcscpy wcscpy [1]	wcsncpy wcsncpy [1]	wcstouq wcstouq [1]
__westol_internal_ westol_internal [1]	mbstowcs mbstowcs [2]	wcscpy wcscpy [1]	wcsncpy wcsncpy [2]	wcswcs wcswcs [2]
__westold_internal_ _westold_internal [1]	mbstowcs mbstowcs [2]	wcrtomb wcrtomb [2]	wcschr wcschr [2]	wcswidth wcswidth [2]
__westoul_internal_ _westoul_internal [1]	mbtowc mbtowc [2]	wcscasecmp wcscasecmp [1]	wcsrtombs wcsrtombs [2]	wcsxfrm wcsxfrm [2]
btowc btowc [2]	putwc putwc [2]	wcscat wcscat [2]	wcsspn wcsspn [2]	wctob wctob [2]
fgetwc fgetwc [2]	putwchar putwchar [2]	wcschr wcschr [2]	wcsstr wcsstr [2]	wctomb wctomb [2]
fgetws fgetws [2]	swprintf swprintf [2]	wcscmp wcscmp [2]	wcstod wcstod [2]	wctrans wctrans [2]
fputwc fputwc [2]	swscanf swscanf [2]	wcscoll wcscoll [2]	wcstof wcstof [2]	wctype wctype [2]

fputws fputws [2]	towctrans towctrans [2]	wesepycscpy [2]	westoimaxwcstoimax [2]	wewidthwcwidth [2]
fwidewide [2]	towlower towlower [2]	wesepnwcscspn [2]	westokwcstok [2]	wmemchrwmemchr [2]
fwprintfwprintf [2]	towupper towupper [2]	wesdupwcscdup [1]	westolwcstol [2]	wmemcpwmemcmp [2]
fwscanfwscanf [2]	ungetwc ungetwc [2]	wesftimewcstime [2]	westoldwcstold [2]	wmemcpywmemcpy [2]
getwc getwc [2]	vfwprintfwprintf [2]	weslenwcslen [2]	westollwcstoll [2]	wmemmovewmemmove [2]
getwchar getwchar [2]	vfwscanfwscanf [2]	wesncasecmpwcncasecmp [1]	westombswcstombs [2]	wmemsetwmemset [2]
mblenmbilen [2]	vswprintfwswprintf [2]	wesncatwcscat [2]	westoqwcstoq [1]	wprintfwprintf [2]
mbrlenmbrlen [2]	vswscanfwswscanf [2]	wesncmpwcscmp [2]	westoulwcstoul [2]	wscanfwscanf [2]
mbrtowcmbrtowc [2]	vwpprintfwprintf [2]	wesnpywcscnp [2]	westoullwcstoull [2]	

Referenced Specification(s)

[1]. Linux Standard Basethis specification

[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3)

1.2.8. String Functions

1.2.8.1. Interfaces for String Functions

An LSB conforming implementation shall provide the generic functions for String Functions specified in Table 1-13, with the full functionality as described in the referenced underlying specification.

Table 1-13. libc - String Functions Function Interfaces

__mempcpy__ mempcpy [1]	bzero bzero [2]	strcasestr strcasestr [1]	strncasecmp strncasecmp [2]	strtoimax strtoimax [2]
__rawmemchr__ rawmemchr [1]	ffs ffs [2]	strcat strcat [2]	strncat strncat [2]	strtok strtok [2]
__stpcpy__ stpcpy [1]	index index [2]	strchr strchr [2]	strncmp strncmp [2]	strtok_r strtok_r [4]2]
__strdup__ strdup [1]	memcpy memcpy [2]	strcmp strcmp [2]	strncpy strncpy [2]	strtol strtol [2]

__strtod_internal__ trtod_internal [1]	memchr memchr [2]	strcoll strcoll [2]	strndup strndup [1]	strtol strtol [2]
__strtof_internal__ trtof_internal [1]	memcmp memcmp [2]	stpncpy stpncpy [2]	strlen strlen [1]	strtoq strtoq [1]
__strtok_r__ strtok_r [1]	memcpy memcpy [2]	strspn strspn [2]	strpbrk strpbrk [2]	strtoull strtoull [2]
__strtol_internal__ trtol_internal [1]	memmove memmove [2]	strdup strdup [2]	strptime strptime [1]	strtoumax strtoumax [2]
__strtold_internal__ strtold_internal [1]	memrchr memrchr [1]	strerror strerror [2]	strchr strchr [2]	strtouq strtouq [1]
__strtoll_internal__ strtoll_internal [1]	memset memset [2]	strerror_r strerror_r [1]	strsep strsep [1]	strverscmp strverscmp [1]
__strtoul_internal__ strtoul_internal [1]	rindex rindex [2]	strfmon strfmon [2]	strsignal strsignal [1]	strxfrm strxfrm [2]
__strtoull_internal__ _strtoull_internal [1]	stpncpy stpncpy [1]	strfry strfry [1]	strspn strspn [2]	swab swab [2]
bcmp bcmp [2]	stpncpy stpncpy [1]	strftime strftime [2]	strstr strstr [2]	
bcopy bcopy [2]	strcasecmp strcasecmp [2]	strlen strlen [2]	strtof strtof [2]	

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

[2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System) and The Single UNIX® Specification (SUS) V3~~

1.2.9. IPC Functions

1.2.9.1. Interfaces for IPC Functions

An LSB conforming implementation shall provide the generic functions for IPC Functions specified in Table 1-14, with the full functionality as described in the referenced underlying specification.

Table 1-14. libc - IPC Functions Function Interfaces

ftok ftok [1]	msgrecv msgrcv [1]	semget semget [1]	shmctl shmctl [1]	
msgctl msgctl [1]	msgsnd msgsnd [1]	semop semop [1]	shmdt shmdt [1]	
msgget msgget [1]	semctl semctl [1]	shmat shmat [1]	shmget shmget [1]	

Referenced Specification(s)

[1]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System) and The Single UNIX® Specification (SUS) V3~~

1.2.10. Regular Expressions

1.2.10.1. Interfaces for Regular Expressions

An LSB conforming implementation shall provide the generic functions for Regular Expressions specified in Table 1-15, with the full functionality as described in the referenced underlying specification.

Table 1-15. libc - Regular Expressions Function Interfaces

regcomp regcomp [1]	regerror regerror [1]	regexec regexec [1]	regfree regfree [1]	
--------------------------------	----------------------------------	--------------------------------	--------------------------------	--

Referenced Specification(s)

[1]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

An LSB conforming implementation shall provide the generic deprecated functions for Regular Expressions specified in Table 1-16, with the full functionality as described in the referenced underlying specification.

These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn in future releases of this specification.

Table 1-16. libc - Regular Expressions Deprecated Function Interfaces

advance advance [1]	re_comp re_comp [1]	re_exec re_exec [1]	step step [1]	
--------------------------------	--------------------------------	--------------------------------	--------------------------	--

Referenced Specification(s)

[1]. ~~CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)~~SUSv2

An LSB conforming implementation shall provide the generic deprecated data interfaces for Regular Expressions specified in Table 1-17, with the full functionality as described in the referenced underlying specification.

These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn in future releases of this specification.

Table 1-17. libc - Regular Expressions Deprecated Data Interfaces

loc1 loc1 [1]	loc2 loc2 [1]	locs locs [1]		
--------------------------	--------------------------	--------------------------	--	--

Referenced Specification(s)

[1]. ~~CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)~~SUSv2

1.2.11. Character Type Functions

1.2.11.1. Interfaces for Character Type Functions

An LSB conforming implementation shall provide the generic functions for Character Type Functions specified in Table 1-18, with the full functionality as described in the referenced underlying specification.

Table 1-18. libc - Character Type Functions Function Interfaces

__ctype_b_loc (GLIBC_2.3) __ctype_b_loc (GLIBC_2.3) [1]	isalpha [2]	ispunct [2]	iswctype [2]	iswupper [2]
__ctype_get_mb_cur_max __ctype_get_mb_cur_max [1]	isascii [2]	isspace [2]	iswdigit [2]	iswxdigit [2]
__ctype_tolower_loc (GLIBC_2.3) __ctype_tolower_loc (GLIBC_2.3) [1]	isctrl [2]	isupper [2]	iswgraph [2]	isxdigit [2]
__ctype_toupper_loc (GLIBC_2.3) __ctype_toupper_loc (GLIBC_2.3) [1]	isdigit [2]	iswalnum [2]	iswlower [2]	toascii [2]
_tolower [2]	isgraph [2]	iswalpha [2]	iswprint [2]	tolower [2]
_toupper [2]	islower [2]	iswblank [2]	iswpunct [2]	toupper [2]
isalnum [2]	isprint [2]	iswcntrl [2]	iswspace [2]	

Referenced Specification(s)

[1]. Linux Standard Base

[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.2.12. Time Manipulation

1.2.12.1. Interfaces for Time Manipulation

An LSB conforming implementation shall provide the generic functions for Time Manipulation specified in Table 1-19, with the full functionality as described in the referenced underlying specification.

Table 1-19. libc - Time Manipulation Function Interfaces

adjtime [1]	etime [2]	gmtime [2]	localtime [1]	alarm [2]
-------------	-----------	------------	---------------	-----------

			e_r [2]	
asctime [2]	etime_r [2]	gmtime_r [2]	mktime [2]	
asctime_r [2]	difftime [2]	localtime [2]	tzset [2]	

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

[2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System) and The Single UNIX® Specification (SUS) V3~~

An LSB conforming implementation shall provide the generic deprecated functions for Time Manipulation specified in Table 1-20, with the full functionality as described in the referenced underlying specification.

These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn in future releases of this specification.

Table 1-20. libc - Time Manipulation Deprecated Function Interfaces

adjtimex [1]				
--------------	--	--	--	--

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

An LSB conforming implementation shall provide the generic data interfaces for Time Manipulation specified in Table 1-21, with the full functionality as described in the referenced underlying specification.

Table 1-21. libc - Time Manipulation Data Interfaces

__daylight__daylight [1]	__tzname__tzname [1]	timezone [2]		
__timezone__timezone [1]	daylight [2]	tzname [2]		

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

[2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System) and The Single UNIX® Specification (SUS) V3~~

1.2.13. Terminal Interface Functions

1.2.13.1. Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the generic functions for Terminal Interface Functions specified in Table 1-22, with the full functionality as described in the referenced underlying specification.

Table 1-22. libc - Terminal Interface Functions Function Interfaces

efgetispeedcfgetispe ed [1]	efsetispeedcfsetispe ed [1]	tedrainsc drain [1]	tegetattrtc getattr [1]	tesendbreaktc sendbr eak [1]
efgetospeedcfgetosp eed [1]	efsetospeedcfsetosp eed [1]	teflowtc flow [1]	tegetpgrptc getpgrp [1]	tesetattrtc setattr [1]
efmakerawcfmakera w [2]	efsetspeedcfsetspee d [2]	teflushc flush [1]	tegetsidtc getsid [1]	tesetpgrptc setpgrp [1]

Referenced Specification(s)

[1]. ISO/IEC 9945: POSIX (2003 Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3)

[2]. Linux Standard Basethis specification

1.2.14. System Database Interface**1.2.14.1. Interfaces for System Database Interface**

An LSB conforming implementation shall provide the generic functions for System Database Interface specified in Table 1-23, with the full functionality as described in the referenced underlying specification.

Table 1-23. libc - System Database Interface Function Interfaces

endgrent endgrent [1]	getgrgid getgrgid [1]	getprotobynumberg etprotobynumber [1]	getservbyport getser vbyport [1]	setgrent setgrent [1]
endnetent endnetent [1]	getgrgid_r getgrgid_ r [1]	getprotoent getproto ent [1]	getservent getservent [1]	setgroups setgroups [2]
endprotoent endprot oent [1]	getgrnam getgrnam [1]	getpwent getpwent [1]	getutent getutent [2]	setnetent setnetent [1]
endpwent endpwent [1]	getgrnam_r getgrna m_r [1]	getpwnam getpwna m [1]	getutent_r getutent_ r [2]	setprotoent setproteo nt [1]
endservent endserve nt [1]	gethostbyaddr getho stbyaddr [1]	getpwnam_r getpwn am_r [1]	getutxent getutxent [1]	setpwent setpwent [1]
endutent endutent [3]	gethostbyname geth ostbyname [1]	getpwuid getpwuid [1]	getutxid getutxid [1]	setservent setservent [1]
endutxent endutxent [1]	getnetbyaddr getnetb yaddr [1]	getpwuid_r getpwui d_r [1]	getutxline getutxline [1]	setutent setutent [2]
getgrent getgrent [1]	getprotobynam getp rotobynam [1]	getservbyname getse rvbyname [1]	pututxline pututxline [1]	setutxent setutxent [1]

Referenced Specification(s)

- [1]. ISO/IEC 9945: POSIX (2003-Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3)
- [2]. ~~Linux Standard Base~~this specification
- [3]. ~~CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0, €606)~~SUSv2

1.2.15. Language Support

1.2.15.1. Interfaces for Language Support

An LSB conforming implementation shall provide the generic functions for Language Support specified in Table 1-24, with the full functionality as described in the referenced underlying specification.

Table 1-24. libc - Language Support Function Interfaces

__libc_start_main__ libc_start_main [1]	__register_atfork(G LIBC_2.3.2) __regis ter_atfork(GLIBC_ 2.3.2) [1]	_obstack_begin_obs tack_begin [1]	_obstack_newchunk _obstack_newchunk [1]	obstack_freeobstack _free [1]
---	---	--	--	--

Referenced Specification(s)

- [1]. ~~Linux Standard Base~~this specification

1.2.16. Large File Support

1.2.16.1. Interfaces for Large File Support

An LSB conforming implementation shall provide the generic functions for Large File Support specified in Table 1-25, with the full functionality as described in the referenced underlying specification.

Table 1-25. libc - Large File Support Function Interfaces

__fxstat64 __fxstat6 4 [1]	fopen64fopen64 [2]	ftello64ftello64 [2]	lseek64lseek64 [2]	readdir64readdir64 [2]
__lxstat64 __lxstat6 4 [1]	freopen64freopen64 [2]	ftruncate64ftruncate 64 [2]	mkstemp64mkstem p64 [2]	statvfs64statvfs64 [2]
__xstat64 __xstat64 [1]	fseeko64fseeko64 [2]	ftw64ftw64 [2]	mmap64mmap64 [2]	tmpfile64tmpfile64 [2]
creat64creat64 [2]	fsetpos64fsetpos64 [2]	getrlimit64getrlimit 64 [2]	nftw64nftw64 [2]	truncate64truncate6 4 [2]
fgetpos64fgetpos64 [2]	fstatvfs64fstatvfs64 [2]	lockf64lockf64 [2]	open64open64 [2]	

Referenced Specification(s)

- [1]. ~~Linux Standard Base~~this specification

217 [2]. Large File Support

1.2.17. Standard Library

218 1.2.17.1. Interfaces for Standard Library

219 An LSB conforming implementation shall provide the generic functions for Standard Library specified in Table 1-26,
220 with the full functionality as described in the referenced underlying specification.

221 **Table 1-26. libc - Standard Library Function Interfaces**

_Exit _Exit [1]	dirname dirname [1]	glob glob [1]	lsearch lsearch [1]	srand srand [1]
__assert_fail __assert_fail [2]	div div [1]	glob64 glob64 [2]	makecontext makecontext [1]	srand48 srand48 [1]
__cxa_atexit __cxa_atexit [2]	drand48 drand48 [1]	globfree globfree [1]	malloc malloc [1]	srandom srandom [1]
__errno_location __errno_location [2]	eevte cv [1]	globfree64 globfree64 [2]	memmem memmem [2]	strtod strtod [1]
__fpending __fpending [2]	erand48 erand48 [1]	grantpt grantpt [1]	mkstemp mkstemp [1]	strtol strtol [1]
__getpagesize __getpagesize [2]	errerr [2]	hcreate hcreate [1]	mktemp mktemp [1]	strtoul strtoul [1]
__isinf __isinf [2]	error error [2]	hdestroy hdestroy [1]	mrnd48 mrnd48 [1]	swapcontext swapcontext [1]
__isinff __isinff [2]	errxerr [2]	hsearch hsearch [1]	nftw nftw [1]	syslog syslog [1]
__isinfl __isinfl [2]	fevtf cv [1]	htonl htonl [1]	nrnd48 nrnd48 [1]	system system [2]
__isnan __isnan [2]	fmtmsg fmtmsg [1]	htons htons [1]	ntohl ntohl [1]	tdelete tdelete [1]
__isnanf __isnanf [2]	fnmatch fnmatch [1]	imaxabs imaxabs [1]	ntohs ntohs [1]	tfind tfind [1]
__isnanl __isnanl [2]	fpathconf fpathconf [1]	imaxdiv imaxdiv [1]	openlog openlog [1]	tmpfile tmpfile [1]
__sysconf __sysconf [2]	free free [1]	inet_addr inet_addr [1]	 perror error [1]	tmpnam tmpnam [1]
_exit _exit [1]	freeaddrinfo freeaddrinfo [1]	inet_ntoa inet_ntoa [1]	posix_memalign posix_memalign [1]	tsearch tsearch [1]
_longjmp _longjmp [1]	ftwlockfile ftwlockfile [1]	inet_ntop inet_ntop [1]	ptsname ptsname [1]	ttname ttname [1]
_setjmp _setjmp [1]	ftw ftw [1]	inet_pton inet_pton [1]	putenv putenv [1]	ttname_r ttname_r [1]
a64l a64l [1]	funlockfile funlockfile [1]	initstate initstate [1]	qsort qsort [1]	twalk twalk [1]

	le [1]			
abort [1]	gai_strerror [1]	insque [1]	rand [1]	unlockpt [1]
abs [1]	getgcvt [1]	isatty [1]	rand_r [1]	unsetenv [1]
atof [1]	getaddrinfo [1]	isblank [1]	random [1]	usleep [1]
atoi [1]	getcwd [1]	jrand48 [1]	random_r [2]	verrx [2]
atol [1]	getdate [1]	l64a [1]	realloc [1]	vfprintf [1]
atoll [1]	getenv [1]	labs [1]	realpath [1]	vscanf [1]
basename [1]	getlogin [1]	lcong48 [1]	remque [1]	vsscanf [1]
bsearch [1]	getnameinfo [1]	ldiv [1]	seed48 [1]	vsyslog [2]
calloc [1]	getopt [2]	lfind [1]	setenv [1]	warn [2]
close [1]	getopt_long [2]	labs [1]	sethostid [2]	warnx [2]
confstr [1]	getopt_long_only [2]	ldiv [1]	sethostname [2]	wordexp [1]
userid [3]	getsubopt [1]	longjmp [1]	setlogmask [1]	wordfree [1]
daemon [2]	gettimeofday [1]	lrand48 [1]	setstate [1]	

Referenced Specification(s)

[1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

[2]. Linux Standard Base this specification

[3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606) SUSv2

An LSB conforming implementation shall provide the generic data interfaces for Standard Library specified in Table 1-27, with the full functionality as described in the referenced underlying specification.

Table 1-27. libc - Standard Library Data Interfaces

__environ [1]	_sys_errlist [1]	getdate_err [2]	opterr [1]	optopt [1]
---------------	------------------	-----------------	------------	------------

_environ <code>_environ</code> [1]	environ <code>environ</code> [2]	optarg <code>optarg</code> [2]	optind <code>optind</code> [1]	
--	---	---	---	--

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

[2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System) and The Single UNIX® Specification (SUS) V3~~

1.3. Data Definitions for libc

This section defines global identifiers and their values that are associated with interfaces contained in libc. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content.

These definitions are intended to supplement those provided in the referenced underlying specifications.

This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are specified in ISO 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.

1.3.1. assert.h

The `assert.h` header shall define the `assert` macro. It refers to the macro `NDEBUG`, which is not defined in this header. If `NDEBUG` is defined before the inclusion of this header, the `assert` macro shall be defined as described below, otherwise the macro shall behave as described in `assert` in ISO/IEC 9945 POSIX.

```
#define assert(expr)    ((void)0)
```

1.3.2. ctype.h

```
enum
{
    _ISupper, _ISlower, _ISalpha, _ISdigit, _ISxdigit, _ISspace, _ISprint,
    _ISgraph, _ISblank, _IScntrl, _ISPunct, _ISalnum
};
```

1.3.3. dirent.h

```
typedef struct __dirstream DIR;

struct dirent
{
    long d_ino;
    off_t d_off;
    unsigned short d_reclen;
```

```

264     unsigned char d_type;
265     char d_name[256];
266 }
267 ;
268 struct dirent64
269 {
270     uint64_t d_ino;
271     int64_t d_off;
272     unsigned short d_reclen;
273     unsigned char d_type;
274     char d_name[256];
275 }
276 ;

```

1.3.4. errno.h

```

277
278 #define errno    (*__errno_location())
279
280 #define EPERM    1
281 #define ECHILD   10
282 #define ENETDOWN 100
283 #define ENETUNREACH 101
284 #define ENETRESET 102
285 #define ECONNABORTED 103
286 #define ECONNRESET 104
287 #define ENOBUFS 105
288 #define EISCONN 106
289 #define ENOTCONN 107
290 #define ESHUTDOWN 108
291 #define ETOOMANYREFS 109
292 #define EAGAIN 11
293 #define ETIMEDOUT 110
294 #define ECONNREFUSED 111
295 #define EHOSTDOWN 112
296 #define EHOSTUNREACH 113
297 #define EALREADY 114
298 #define EINPROGRESS 115
299 #define ESTALE 116
300 #define EUCLEAN 117
301 #define ENOTNAM 118
302 #define ENAVAIL 119
303 #define ENOMEM 12
304 #define EISNAM 120
305 #define EREMOTEIO 121
306 #define EDQUOT 122
307 #define ENOMEDIUM 123
308 #define EMEDIUMTYPE 124
309 #define ECANCELED 125
310 #define EACCES 13
311 #define EFAULT 14
312 #define ENOTBLK 15

```

```

313 #define EBUSY 16
314 #define EEXIST 17
315 #define EXDEV 18
316 #define ENODEV 19
317 #define ENOENT 2
318 #define ENOTDIR 20
319 #define EISDIR 21
320 #define EINVAL 22
321 #define ENFILE 23
322 #define EMFILE 24
323 #define ENOTTY 25
324 #define ETXTBSY 26
325 #define EFBIG 27
326 #define ENOSPC 28
327 #define ESPIPE 29
328 #define ESRCH 3
329 #define EROFS 30
330 #define EMLINK 31
331 #define EPIPE 32
332 #define EDOM 33
333 #define ERANGE 34
334 #define EDEADLK 35
335 #define ENAMETOOLONG 36
336 #define ENOLCK 37
337 #define ENOSYS 38
338 #define ENOTEMPTY 39
339 #define EINTR 4
340 #define ELOOP 40
341 #define ENOMSG 42
342 #define EIDRM 43
343 #define ECHRNG 44
344 #define EL2NSYNC 45
345 #define EL3HLT 46
346 #define EL3RST 47
347 #define ELNRNG 48
348 #define EUNATCH 49
349 #define EIO 5
350 #define ENOANO 55
351 #define EBADRQC 56
352 #define EBADSLT 57
353 #define EBFONT 59
354 #define ENXIO 6
355 #define ENOSTR 60
356 #define ENODATA 61
357 #define ETIME 62
358 #define ENOSR 63
359 #define ENONET 64
360 #define ENOPKG 65
361 #define EREMOTE 66
362 #define ENOLINK 67
363 #define EADV 68
364 #define ESRMNT 69
365 #define E2BIG 7

```

```

366 #define ECOMM      70
367 #define EPROTO     71
368 #define EMULTIHOP   72
369 #define EDOTDOT     73
370 #define EBADMSG     74
371 #define EOVERFLOW   75
372 #define ENOTUNIQ    76
373 #define EBADFD      77
374 #define EREMCHG     78
375 #define ELIBACC     79
376 #define ENOEXEC     8
377 #define ELIBBAD     80
378 #define ELIBSCN     81
379 #define ELIBMAX     82
380 #define ELIBEXEC    83
381 #define EILSEQ      84
382 #define ERESTART    85
383 #define ESTRPIPE    86
384 #define EUSERS      87
385 #define ENOTSOCK    88
386 #define EDESTADDRREQ 89
387 #define EBADF       9
388 #define EMSGSIZE    90
389 #define EPROTOTYPE  91
390 #define ENOPROTOOPT 92
391 #define EPROTONOSUPPORT 93
392 #define ESOCKTNOSUPPORT 94
393 #define EOPNOTSUPP   95
394 #define EPNOSUPPORT  96
395 #define EAFNOSUPPORT 97
396 #define EADDRINUSE   98
397 #define EADDRNOTAVAIL 99
398 #define EWOULDBLOCK  EAGAIN
399 #define ENOTSUP      EOPNOTSUPP

```

1.3.5.fcntl.h

```

400
401 #define O_RDONLY      00
402 #define O_ACCMODE     0003
403 #define O_WRONLY      01
404 #define O_CREAT       0100
405 #define O_TRUNC       01000
406 #define O_SYNC        010000
407 #define O_RDWR       02
408 #define O_EXCL        0200
409 #define O_APPEND      02000
410 #define O_ASYNC       020000
411 #define O_NOCTTY      0400
412 #define O_NDELAY      04000
413 #define O_NONBLOCK    04000
414 #define FD_CLOEXEC    1

```

```

415
416 struct flock
417 {
418     short l_type;
419     short l_whence;
420     off_t l_start;
421     off_t l_len;
422     pid_t l_pid;
423 }
424 ;
425 struct flock64
426 {
427     short l_type;
428     short l_whence;
429     loff_t l_start;
430     loff_t l_len;
431     pid_t l_pid;
432 }
433 ;
434
435 #define F_DUPFD 0
436 #define F_RDLCK 0
437 #define F_GETFD 1
438 #define F_WRLCK 1
439 #define F_SETFD 2
440 #define F_UNLCK 2
441 #define F_GETFL 3
442 #define F_SETFL 4
443 #define F_GETLK 5
444 #define F_SETLK 6
445 #define F_SETLKW      7
446 #define F_SETOWN      8
447 #define F_GETOWN      9

```

1.3.6. fmtmsg.h

```

448
449 #define MM_HARD 1
450 #define MM_NRECOV      128
451 #define MM_UTIL 16
452 #define MM_SOFT 2
453 #define MM_OPSYS      32
454 #define MM_FIRM 4
455 #define MM_RECOVER      64
456 #define MM_APPL 8
457
458 #define MM_NOSEV      0
459 #define MM_HALT 1
460 #define MM_ERROR      2
461
462 #define MM_NULLLBL      ((char *) 0)

```

1.3.7. fnmatch.h

```

463
464 #define FNM_PATHNAME      (1<<0)
465 #define FNM_NOESCAPE     (1<<1)
466 #define FNM_PERIOD       (1<<2)
467 #define FNM_NOMATCH      1

```

1.3.8. ftw.h

```

468
469 #define FTW_D      FTW_D
470 #define FTW_DNR   FTW_DNR
471 #define FTW_DP    FTW_DP
472 #define FTW_F     FTW_F
473 #define FTW_NS    FTW_NS
474 #define FTW_SL    FTW_SL
475 #define FTW_SLN   FTW_SLN
476
477 enum
478 {
479     FTW_F, FTW_D, FTW_DNR, FTW_NS, FTW_SL, FTW_DP, FTW_SLN
480 }
481 ;
482
483 enum
484 {
485     FTW_PHYS, FTW_MOUNT, FTW_CHDIR, FTW_DEPTH
486 }
487 ;
488
489 struct FTW
490 {
491     int base;
492     int level;
493 }
494 ;
495
496 typedef int (*__ftw_func_t) (char *__filename, struct stat * __status,
497                             int __flag);
498 typedef int (*__ftw64_func_t) (char *__filename, struct stat64 * __status,
499                               int __flag);
500 typedef int (*__nftw_func_t) (char *__filename, struct stat * __status,
501                               int __flag, struct FTW * __info);
502 typedef int (*__nftw64_func_t) (char *__filename, struct stat64 * __status,
503                                 int __flag, struct FTW * __info);

```

1.3.9. getopt.h

```

504
505 #define no_argument      0
506 #define required_argument 1

```

```

507 #define optional_argument      2
508
509 struct option
510 {
511     char *name;
512     int has_arg;
513     int *flag;
514     int val;
515 }
516 ;

```

1.3.10. glob.h

```

517
518 #define GLOB_ERR      (1<<0)
519 #define GLOB_MARK     (1<<1)
520 #define GLOB_BRACE    (1<<10)
521 #define GLOB_NOMAGIC  (1<<11)
522 #define GLOB_TILDE    (1<<12)
523 #define GLOB_ONLYDIR  (1<<13)
524 #define GLOB_TILDE_CHECK (1<<14)
525 #define GLOB_NOSORT   (1<<2)
526 #define GLOB_DOOFFS   (1<<3)
527 #define GLOB_NOCHECK  (1<<4)
528 #define GLOB_APPEND   (1<<5)
529 #define GLOB_NOESCAPE (1<<6)
530 #define GLOB_PERIOD   (1<<7)
531 #define GLOB_MAGCHAR  (1<<8)
532 #define GLOB_ALTDIRFUNC (1<<9)
533
534 #define GLOB_NOSPACE    1
535 #define GLOB_ABORTED    2
536 #define GLOB_NOMATCH    3
537 #define GLOB_NOSYS      4
538
539 typedef struct
540 {
541     size_t gl_pathc;
542     char **gl_pathv;
543     size_t gl_offs;
544     int gl_flags;
545     void (*gl_closedir) (void *);
546     struct dirent *(*gl_readdir) (void *);
547     void *(*gl_opendir) (const char *);
548     int (*gl_lstat) (const char *, struct stat *);
549     int (*gl_stat) (const char *, struct stat *);
550 }
551 glob_t;
552
553 typedef struct
554 {
555     size_t gl_pathc;

```



```

556     char **gl_pathv;
557     size_t gl_offs;
558     int gl_flags;
559     void (*gl_closedir) (void *);
560     struct dirent64 *(*gl_readdir64) (void *);
561     void (*gl_opendir) (const char *);
562     int (*gl_lstat) (const char *, struct stat *);
563     int (*gl_stat) (const char *, struct stat *);
564 }
565 glob64_t;

```

1.3.11. grp.h

```

566
567 struct group
568 {
569     char *gr_name;
570     char *gr_passwd;
571     gid_t gr_gid;
572     char **gr_mem;
573 }
574 ;

```

1.3.12. iconv.h

```

575
576 typedef void *iconv_t;

```

1.3.13. inttypes.h

```

577
578 typedef lldiv_t imaxdiv_t;
579 typedef unsigned char uint8_t;
580 typedef unsigned short uint16_t;
581 typedef unsigned int uint32_t;

```

1.3.14. langinfo.h

```

582
583 #define ABDAY_1 0x20000
584 #define ABDAY_2 0x20001
585 #define ABDAY_3 0x20002
586 #define ABDAY_4 0x20003
587 #define ABDAY_5 0x20004
588 #define ABDAY_6 0x20005
589 #define ABDAY_7 0x20006
590
591 #define DAY_1 0x20007
592 #define DAY_2 0x20008
593 #define DAY_3 0x20009
594 #define DAY_4 0x2000A

```

```

595 #define DAY_5      0x2000B
596 #define DAY_6      0x2000C
597 #define DAY_7      0x2000D
598
599 #define ABMON_1     0x2000E
600 #define ABMON_2     0x2000F
601 #define ABMON_3     0x20010
602 #define ABMON_4     0x20011
603 #define ABMON_5     0x20012
604 #define ABMON_6     0x20013
605 #define ABMON_7     0x20014
606 #define ABMON_8     0x20015
607 #define ABMON_9     0x20016
608 #define ABMON_10    0x20017
609 #define ABMON_11    0x20018
610 #define ABMON_12    0x20019
611
612 #define MON_1       0x2001A
613 #define MON_2       0x2001B
614 #define MON_3       0x2001C
615 #define MON_4       0x2001D
616 #define MON_5       0x2001E
617 #define MON_6       0x2001F
618 #define MON_7       0x20020
619 #define MON_8       0x20021
620 #define MON_9       0x20022
621 #define MON_10      0x20023
622 #define MON_11      0x20024
623 #define MON_12      0x20025
624
625 #define AM_STR      0x20026
626 #define PM_STR      0x20027
627
628 #define D_T_FMT     0x20028
629 #define D_FMT       0x20029
630 #define T_FMT       0x2002A
631 #define T_FMT_AMPM  0x2002B
632
633 #define ERA          0x2002C
634 #define ERA_D_FMT    0x2002E
635 #define ALT_DIGITS   0x2002F
636 #define ERA_D_T_FMT  0x20030
637 #define ERA_T_FMT    0x20031
638
639 #define CODESET 14
640
641 #define CRNCYSTR     0x4000F
642
643 #define RADIXCHAR     0x10000
644 #define THOUSEP       0x10001
645 #define YESEXPR       0x50000
646 #define NOEXPR        0x50001
647 #define YESSTR        0x50002

```

```
648 #define NOSTR    0x50003
```

1.3.15. limits.h

```
649
650 #define LLONG_MIN      (-LLONG_MAX-1LL)
651 #define ULLONG_MAX     18446744073709551615ULL
652 #define OPEN_MAX      256
653 #define PATH_MAX      4096
654 #define LLONG_MAX     9223372036854775807LL
655 #define SSIZE_MAX     LONG_MAX
656
657 #define MB_LEN_MAX     16
658
659 #define SCHAR_MIN      (-128)
660 #define SCHAR_MAX     127
661 #define UCHAR_MAX     255
662 #define CHAR_BIT       8
663
664 #define SHRT_MIN       (-32768)
665 #define SHRT_MAX      32767
666 #define USHRT_MAX     65535
667
668 #define INT_MIN        (-INT_MAX-1)
669 #define INT_MAX      2147483647
670 #define __INT_MAX__    2147483647
671 #define UINT_MAX      4294967295U
672
673 #define LONG_MIN       (-LONG_MAX-1L)
```

1.3.16. locale.h

```
674
675 #define LC_CTYPE       0
676 #define LC_NUMERIC     1
677 #define LC_TELEPHONE   10
678 #define LC_MEASUREMENT 11
679 #define LC_IDENTIFICATION 12
680 #define LC_TIME        2
681 #define LC_COLLATE     3
682 #define LC_MONETARY    4
683 #define LC_MESSAGES    5
684 #define LC_ALL         6
685 #define LC_PAPER       7
686 #define LC_NAME        8
687 #define LC_ADDRESS     9
688
689 struct lconv
690 {
691     char *decimal_point;
692     char *thousands_sep;
693     char *grouping;
```

```

694     char *int_curr_symbol;
695     char *currency_symbol;
696     char *mon_decimal_point;
697     char *mon_thousands_sep;
698     char *mon_grouping;
699     char *positive_sign;
700     char *negative_sign;
701     char int_frac_digits;
702     char frac_digits;
703     char p_cs_precedes;
704     char p_sep_by_space;
705     char n_cs_precedes;
706     char n_sep_by_space;
707     char p_sign_posn;
708     char n_sign_posn;
709     char int_p_cs_precedes;
710     char int_p_sep_by_space;
711     char int_n_cs_precedes;
712     char int_n_sep_by_space;
713     char int_p_sign_posn;
714     char int_n_sign_posn;
715 }
716 ;
717
718 typedef struct __locale_struct
719 {
720     struct locale_data *__locales[13];
721     const unsigned short *__ctype_b;
722     const int *__ctype_tolower;
723     const int *__ctype_toupper;
724     const char *__names[13];
725 }
726 *__locale_t;

```

1.3.17. net/if.h

```

727
728 #define IF_NAMESIZE      16
729
730 #define IFF_UP           0x01
731 #define IFF_BROADCAST    0x02
732 #define IFF_DEBUG        0x04
733 #define IFF_LOOPBACK     0x08
734 #define IFF_POINTOPOINT  0x10
735 #define IFF_PROMISC      0x100
736 #define IFF_MULTICAST    0x1000
737 #define IFF_NOTRAILERS   0x20
738 #define IFF_RUNNING      0x40
739 #define IFF_NOARP        0x80
740
741 struct ifaddr
742 {

```

```

743     struct sockaddr ifa_addr;
744     union
745     {
746         struct sockaddr ifu_broadaddr;
747         struct sockaddr ifu_dstaddr;
748     }
749     ifa_ifu;
750     void *ifa_ifp;
751     void *ifa_next;
752 }
753 ;
754 #define IFNAMSIZ          IF_NAMESIZE
755
756 struct ifreq
757 {
758     union
759     {
760         char ifrn_name[IFNAMSIZ];
761     }
762     ifr_ifrn;
763     union
764     {
765         struct sockaddr ifru_addr;
766         struct sockaddr ifru_dstaddr;
767         struct sockaddr ifru_broadaddr;
768         struct sockaddr ifru_netmask;
769         struct sockaddr ifru_hwaddr;
770         short ifru_flags;
771         int ifru_ivalue;
772         int ifru_mtu;
773         char ifru_slave[IFNAMSIZ];
774         char ifru_newname[IFNAMSIZ];
775         caddr_t ifru_data;
776         struct ifmap ifru_map;
777     }
778     ifr_ifru;
779 }
780 ;
781
782 struct ifconf
783 {
784     int ifc_len;
785     union
786     {
787         caddr_t ifcu_buf;
788         struct ifreq *ifcu_req;
789     }
790     ifc_ifcu;
791 }
792 ;

```

1.3.18. netdb.h

```

793
794 #define h_errno (*__h_errno_location ())
795 #define NETDB_INTERNAL -1
796 #define NETDB_SUCCESS 0
797 #define HOST_NOT_FOUND 1
798 #define IPPORT_RESERVED 1024
799 #define NI_MAXHOST 1025
800 #define TRY_AGAIN 2
801 #define NO_RECOVERY 3
802 #define NI_MAXSERV 32
803 #define NO_DATA 4
804 #define h_addr h_addr_list[0]
805 #define NO_ADDRESS NO_DATA
806
807 struct servent
808 {
809     char *s_name;
810     char **s_aliases;
811     int s_port;
812     char *s_proto;
813 }
814 ;
815 struct hostent
816 {
817     char *h_name;
818     char **h_aliases;
819     int h_addrtype;
820     int h_length;
821     char **h_addr_list;
822 }
823 ;
824 struct protoent
825 {
826     char *p_name;
827     char **p_aliases;
828     int p_proto;
829 }
830 ;
831 struct netent
832 {
833     char *n_name;
834     char **n_aliases;
835     int n_addrtype;
836     unsigned int n_net;
837 }
838 ;
839 #define AI_PASSIVE 0x0001
840 #define AI_CANONNAME 0x0002
841 #define AI_NUMERICHOST 0x0004
842

```

```

843 struct addrinfo
844 {
845     int ai_flags;
846     int ai_family;
847     int ai_socktype;
848     int ai_protocol;
849     socklen_t ai_addrlen;
850     struct sockaddr *ai_addr;
851     char *ai_canonname;
852     struct addrinfo *ai_next;
853 }
854 ;
855 #define NI_NUMERICHOST 1
856 #define NI_DGRAM 16
857 #define NI_NUMERICSERV 2
858 #define NI_NOFQDN 4
859 #define NI_NAMEREQD 8
860
861 #define EAI_BADFLAGS -1
862 #define EAI_MEMORY -10
863 #define EAI_SYSTEM -11
864 #define EAI_NONAME -2
865 #define EAI_AGAIN -3
866 #define EAI_FAIL -4
867 #define EAI_NODATA -5
868 #define EAI_FAMILY -6
869 #define EAI_SOCKTYPE -7
870 #define EAI_SERVICE -8
871 #define EAI_ADDRFAMILY -9

```

1.3.19. netinet/in.h

```

872
873 #define IPPROTO_IP 0
874 #define IPPROTO_ICMP 1
875 #define IPPROTO_UDP 17
876 #define IPPROTO_IGMP 2
877 #define IPPROTO_RAW 255
878 #define IPPROTO_IPV6 41
879 #define IPPROTO_ICMPV6 58
880 #define IPPROTO_TCP 6
881
882 typedef uint16_t in_port_t;
883
884 struct in_addr
885 {
886     uint32_t s_addr;
887 }
888 ;
889 typedef uint32_t in_addr_t;
890 #define INADDR_NONE ((in_addr_t) 0xffffffff)
891 #define INADDR_BROADCAST (0xffffffff)

```

```

892 #define INADDR_ANY      0
893
894 struct in6_addr
895 {
896     union
897     {
898         uint8_t u6_addr8[16];
899         uint16_t u6_addr16[8];
900         uint32_t u6_addr32[4];
901     }
902     in6_u;
903 }
904 ;
905 #define IN6ADDR_ANY_INIT    { { { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } } }
906 #define IN6ADDR_LOOPBACK_INIT { { { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 } } }
907
908 #define INET_ADDRSTRLEN 16
909
910 struct sockaddr_in
911 {
912     sa_family_t sin_family;
913     unsigned short sin_port;
914     struct in_addr sin_addr;
915     unsigned char sin_zero[8];
916 }
917 ;
918 #define INET6_ADDRSTRLEN    46
919
920 struct sockaddr_in6
921 {
922     unsigned short sin6_family;
923     uint16_t sin6_port;
924     uint32_t sin6_flowinfo;
925     struct in6_addr sin6_addr;
926     uint32_t sin6_scope_id;
927 }
928 ;
929 #define SOL_IP  0
930 #define IP_TOS  1
931 #define IPV6_UNICAST_HOPS    16
932 #define IPV6_MULTICAST_IF    17
933 #define IPV6_MULTICAST_HOPS  18
934 #define IPV6_MULTICAST_LOOP  19
935 #define IPV6_JOIN_GROUP      20
936 #define IPV6_LEAVE_GROUP     21
937 #define IPV6_V6ONLY          26
938 #define IP_MULTICAST_IF      32
939 #define IP_MULTICAST_TTL     33
940 #define IP_MULTICAST_LOOP    34
941 #define IP_ADD_MEMBERSHIP    35
942 #define IP_DROP_MEMBERSHIP   36
943
944 struct ipv6_mreq

```



```

945 {
946     struct in6_addr ipv6mr_multiaddr;
947     int ipv6mr_interface;
948 }
949 ;
950 struct ip_mreq
951 {
952     struct in_addr imr_multiaddr;
953     struct in_addr imr_interface;
954 }
955 ;

```

1.3.20. netinet/tcp.h

```

956
957 #define TCP_NODELAY    1
958 #define SOL_TCP 6

```

1.3.21. netinet/udp.h

```

959
960 #define SOL_UDP 17

```

1.3.22. nl_types.h

```

961
962 #define NL_CAT_LOCALE    1
963 #define NL_SETD 1
964
965 typedef void *nl_catd;
966
967 typedef int nl_item;

```

1.3.23. pty.h

```

968
969 struct winsize
970 {
971     unsigned short ws_row;
972     unsigned short ws_col;
973     unsigned short ws_xpixel;
974     unsigned short ws_ypixel;
975 }
976 ;

```

1.3.24. pwd.h

```

977
978 struct passwd
979 {
980     char *pw_name;

```

```

981     char *pw_passwd;
982     uid_t pw_uid;
983     gid_t pw_gid;
984     char *pw_gecos;
985     char *pw_dir;
986     char *pw_shell;
987 }
988 ;

```

1.3.25. regex.h

```

989
990 #define RE_BACKSLASH_ESCAPE_IN_LISTS    ((unsigned long int)1)
991 #define RE_BK_PLUS_QM    (RE_BACKSLASH_ESCAPE_IN_LISTS<<1)
992 #define RE_SYNTAX_AWK    (RE_BACKSLASH_ESCAPE_IN_LISTS|RE_DOT_NOT_NULL|RE_NO_BK_PARENS|
993 RE_NO_BK_REFS| RE_NO_BK_VBAR| RE_NO_EMPTY_RANGES| RE_DOT_NEWLINE|
994 RE_CONTEXT_INDEP_ANCHORS| RE_UNMATCHED_RIGHT_PAREN_ORD | RE_NO_GNU_OPS)
995 #define RE_CHAR_CLASSES (RE_BK_PLUS_QM<<1)
996 #define RE_SYNTAX_GREP
997 (RE_BK_PLUS_QM|RE_CHAR_CLASSES|RE_HAT_LISTS_NOT_NEWLINE|RE_INTERVALS|RE_NEWLINE_ALT)
998 #define RE_CONTEXT_INDEP_ANCHORS    (RE_CHAR_CLASSES<<1)
999 #define RE_SYNTAX_EGREP (RE_CHAR_CLASSES|RE_CONTEXT_INDEP_ANCHORS|
1000 RE_CONTEXT_INDEP_OPS|RE_HAT_LISTS_NOT_NEWLINE|RE_NEWLINE_ALT|RE_NO_BK_PARENS|RE_NO_BK_
1001 VBAR)
1002 #define _RE_SYNTAX_POSIX_COMMON
1003 (RE_CHAR_CLASSES|RE_DOT_NEWLINE|RE_DOT_NOT_NULL|RE_INTERVALS|RE_NO_EMPTY_RANGES)
1004 #define RE_CONTEXT_INDEP_OPS    (RE_CONTEXT_INDEP_ANCHORS<<1)
1005 #define RE_CONTEXT_INVALID_OPS (RE_CONTEXT_INDEP_OPS<<1)
1006 #define RE_DOT_NEWLINE    (RE_CONTEXT_INVALID_OPS<<1)
1007 #define RE_INVALID_INTERVAL_ORD (RE_DEBUG<<1)
1008 #define RE_DOT_NOT_NULL (RE_DOT_NEWLINE<<1)
1009 #define RE_HAT_LISTS_NOT_NEWLINE    (RE_DOT_NOT_NULL<<1)
1010 #define RE_INTERVALS    (RE_HAT_LISTS_NOT_NEWLINE<<1)
1011 #define RE_LIMITED_OPS    (RE_INTERVALS<<1)
1012 #define RE_NEWLINE_ALT    (RE_LIMITED_OPS<<1)
1013 #define RE_NO_BK_BRACES (RE_NEWLINE_ALT<<1)
1014 #define RE_NO_BK_PARENS (RE_NO_BK_BRACES<<1)
1015 #define RE_NO_BK_REFS    (RE_NO_BK_PARENS<<1)
1016 #define RE_NO_BK_VBAR    (RE_NO_BK_REFS<<1)
1017 #define RE_NO_EMPTY_RANGES    (RE_NO_BK_VBAR<<1)
1018 #define RE_UNMATCHED_RIGHT_PAREN_ORD    (RE_NO_EMPTY_RANGES<<1)
1019 #define RE_DEBUG    (RE_NO_GNU_OPS<<1)
1020 #define RE_NO_GNU_OPS    (RE_NO_POSIX_BACKTRACKING<<1)
1021 #define RE_SYNTAX_POSIX_EGREP
1022 (RE_SYNTAX_EGREP|RE_INTERVALS|RE_NO_BK_BRACES|RE_INVALID_INTERVAL_ORD)
1023 #define RE_SYNTAX_POSIX_AWK
1024 (RE_SYNTAX_POSIX_EXTENDED|RE_BACKSLASH_ESCAPE_IN_LISTS|RE_INTERVALS|RE_NO_GNU_OPS)
1025 #define RE_NO_POSIX_BACKTRACKING    (RE_UNMATCHED_RIGHT_PAREN_ORD<<1)
1026 #define RE_SYNTAX_POSIX_BASIC    (_RE_SYNTAX_POSIX_COMMON|RE_BK_PLUS_QM)
1027 #define RE_SYNTAX_POSIX_EXTENDED
1028 (_RE_SYNTAX_POSIX_COMMON|RE_CONTEXT_INDEP_ANCHORS|RE_CONTEXT_INDEP_OPS|RE_NO_BK_BRACES
1029 |RE_NO_BK_PARENS|RE_NO_BK_VBAR|RE_CONTEXT_INVALID_OPS|RE_UNMATCHED_RIGHT_PAREN_ORD)

```

```

1030 #define RE_SYNTAX_POSIX_MINIMAL_EXTENDED
1031 (_RE_SYNTAX_POSIX_COMMON|RE_CONTEXT_INDEP_ANCHORS|RE_CONTEXT_INVALID_OPS|RE_NO_BK_BRAC
1032 ES|RE_NO_BK_PARENS|RE_NO_BK_REFS|RE_NO_BK_VBAR|RE_UNMATCHED_RIGHT_PAREN_ORD)
1033 #define RE_SYNTAX_POSIX_MINIMAL_BASIC (_RE_SYNTAX_POSIX_COMMON|RE_LIMITED_OPS)
1034 #define RE_SYNTAX_ED RE_SYNTAX_POSIX_BASIC
1035 #define RE_SYNTAX_SED RE_SYNTAX_POSIX_BASIC
1036
1037 typedef unsigned long reg_syntax_t;
1038
1039 typedef struct re_pattern_buffer
1040 {
1041     unsigned char *buffer;
1042     unsigned long allocated;
1043     unsigned long used;
1044     reg_syntax_t syntax;
1045     char *fastmap;
1046     char *translate;
1047     size_t re_nsub;
1048     unsigned int can_be_null:1;
1049     unsigned int regs_allocated:2;
1050     unsigned int fastmap_accurate:1;
1051     unsigned int no_sub:1;
1052     unsigned int not_bol:1;
1053     unsigned int not_eol:1;
1054     unsigned int newline_anchor:1;
1055 }
1056 regex_t;
1057 typedef int regoff_t;
1058 typedef struct
1059 {
1060     regoff_t rm_so;
1061     regoff_t rm_eo;
1062 }
1063 regmatch_t;
1064 #define REG_NOTEOL (1<<1)
1065 #define REG_ICASE (REG_EXTENDED<<1)
1066 #define REG_NEWLINE (REG_ICASE<<1)
1067 #define REG_NOSUB (REG_NEWLINE<<1)
1068 #define REG_NOMATCH -1
1069 #define REG_EXTENDED 1
1070 #define REG_NOTBOL 1

```

1.3.26. rpc/auth.h

```

1071
1072 enum auth_stat
1073 {
1074     AUTH_OK, AUTH_BADCRED = 1, AUTH_REJECTEDCRED = 2, AUTH_BADVERF =
1075     3, AUTH_REJECTEDVERF = 4, AUTH_TOOWEAK = 5, AUTH_INVALIDRESP =
1076     6, AUTH_FAILED = 7
1077 }
1078 ;

```

```

1079
1080 union des_block
1081 {
1082     struct
1083     {
1084         u_int32_t high;
1085         u_int32_t low;
1086     }
1087     key;
1088     char c[8];
1089 }
1090 ;
1091
1092 struct opaque_auth
1093 {
1094     enum_t oa_flavor;
1095     caddr_t oa_base;
1096     u_int oa_length;
1097 }
1098 ;
1099
1100 typedef struct AUTH
1101 {
1102     struct opaque_auth ah_cred;
1103     struct opaque_auth ah_verf;
1104     union des_block ah_key;
1105     struct auth_ops *ah_ops;
1106     caddr_t ah_private;
1107 }
1108 AUTH;
1109
1110 struct auth_ops
1111 {
1112     void (*ah_nextverf) (struct AUTH *);
1113     int (*ah_marshall) (struct AUTH *, XDR *);
1114     int (*ah_validate) (struct AUTH *, struct opaque_auth *);
1115     int (*ah_refresh) (struct AUTH *);
1116     void (*ah_destroy) (struct AUTH *);
1117 }
1118 ;

```

1.3.27. rpc/clnt.h

```

1119
1120 #define clnt_control(cl,rq,in) ((*(cl)->cl_ops->cl_control)(cl,rq,in))
1121 #define clnt_abort(rh) ((*(rh)->cl_ops->cl_abort)(rh))
1122 #define clnt_call(rh, proc, xargs, argsp, xres, resp, secs) ((*(rh)->cl_ops->cl_call)(rh,
1123 proc, xargs, argsp, xres, resp, secs))
1124 #define clnt_destroy(rh) ((*(rh)->cl_ops->cl_destroy)(rh))
1125 #define clnt_freeres(rh,xres,resp) ((*(rh)->cl_ops->cl_freeres)(rh,xres,resp))
1126 #define clnt_geterr(rh,errp) ((*(rh)->cl_ops->cl_geterr)(rh, errp))
1127 #define NULLPROC ((u_long)0)

```

```

1128 #define CLSET_TIMEOUT      1
1129 #define CLGET_XID           10
1130 #define CLSET_XID           11
1131 #define CLGET_VERS          12
1132 #define CLSET_VERS          13
1133 #define CLGET_PROG          14
1134 #define CLSET_PROG          15
1135 #define CLGET_TIMEOUT       2
1136 #define CLGET_SERVER_ADDR   3
1137 #define CLSET_RETRY_TIMEOUT  4
1138 #define CLGET_RETRY_TIMEOUT  5
1139 #define CLGET_FD            6
1140 #define CLGET_SVC_ADDR      7
1141 #define CLSET_FD_CLOSE      8
1142 #define CLSET_FD_NCLOSE     9
1143
1144 enum clnt_stat
1145 {
1146     RPC_SUCCESS, RPC_CANTENCODEARGS = 1, RPC_CANTDECODERES = 2, RPC_CANTSEND =
1147     3, RPC_CANTRECV = 4, RPC_TIMEDOUT = 5, RPC_VERSIONMISMATCH =
1148     6, RPC_AUTHERROR = 7, RPC_PROGUNAVAIL = 8, RPC_PROGVERSIONMISMATCH =
1149     9, RPC_PROGUNAVAIL = 10, RPC_CANTDECODEARGS = 11, RPC_SYSTEMERROR =
1150     12, RPC_NOBROADCAST = 21, RPC_UNKNOWNHOST = 13, RPC_UNKNOWNPROTO =
1151     17, RPC_UNKNOWNADDR = 19, RPC_RPCBFAILURE = 14, RPC_PROGNOTREGISTERED =
1152     15, RPC_N2AXLATEFAILURE = 22, RPC_FAILED = 16, RPC_INTR =
1153     18, RPC_TLIERROR = 20, RPC_UDERROR = 23, RPC_INPROGRESS =
1154     24, RPC_STALERACHANDLE = 25
1155 }
1156 ;
1157 struct rpc_err
1158 {
1159     enum clnt_stat re_status;
1160     union
1161     {
1162         int RE_errno;
1163         enum auth_stat RE_why;
1164         struct
1165         {
1166             u_long low;
1167             u_long high;
1168         }
1169         RE_vers;
1170         struct
1171         {
1172             long s1;
1173             long s2;
1174         }
1175         RE_lb;
1176     }
1177     ru;
1178 }
1179 ;
1180

```

```

1181 typedef struct CLIENT
1182 {
1183     struct AUTH *cl_auth;
1184     struct clnt_ops *cl_ops;
1185     caddr_t cl_private;
1186 }
1187 CLIENT;
1188
1189 struct clnt_ops
1190 {
1191     enum clnt_stat (*cl_call) (struct CLIENT *, u_long, xdrproc_t, caddr_t,
1192                               xdrproc_t, caddr_t, struct timeval);
1193     void (*cl_abort) (void);
1194     void (*cl_geterr) (struct CLIENT *, struct rpc_err *);
1195     bool_t (*cl_freeres) (struct CLIENT *, xdrproc_t, caddr_t);
1196     void (*cl_destroy) (struct CLIENT *);
1197     bool_t (*cl_control) (struct CLIENT *, int, char *);
1198 }
1199 ;

```

1.3.28. rpc/rpc_msg.h

```

1200
1201 enum msg_type
1202 {
1203     CALL, REPLY = 1
1204 }
1205 ;
1206
1207 enum reply_stat
1208 {
1209     MSG_ACCEPTED, MSG_DENIED = 1
1210 }
1211 ;
1212
1213 enum accept_stat
1214 {
1215     SUCCESS, PROG_UNAVAIL = 1, PROG_MISMATCH = 2, PROC_UNAVAIL =
1216     3, GARBAGE_ARGS = 4, SYSTEM_ERR = 5
1217 }
1218 ;
1219
1220 enum reject_stat
1221 {
1222     RPC_MISMATCH, AUTH_ERROR = 1
1223 }
1224 ;
1225
1226 struct accepted_reply
1227 {
1228     struct opaque_auth ar_verf;
1229     enum accept_stat ar_stat;
1230     union
1231     {
1232         struct

```

```

1230     {
1231         unsigned long low;
1232         unsigned long high;
1233     }
1234     AR_versions;
1235     struct
1236     {
1237         caddr_t where;
1238         xdrproc_t proc;
1239     }
1240     AR_results;
1241 }
1242 ru;
1243 }
1244 ;
1245
1246 struct rejected_reply
1247 {
1248     enum reject_stat rj_stat;
1249     union
1250     {
1251         struct
1252         {
1253             unsigned long low;
1254             unsigned long high;
1255         }
1256         RJ_versions;
1257         enum auth_stat RJ_why;
1258     }
1259     ru;
1260 }
1261 ;
1262
1263 struct reply_body
1264 {
1265     enum reply_stat rp_stat;
1266     union
1267     {
1268         struct accepted_reply RP_ar;
1269         struct rejected_reply RP_dr;
1270     }
1271     ru;
1272 }
1273 ;
1274
1275 struct call_body
1276 {
1277     unsigned long cb_rpcvers;
1278     unsigned long cb_prog;
1279     unsigned long cb_vers;
1280     unsigned long cb_proc;
1281     struct opaque_auth cb_cred;
1282     struct opaque_auth cb_verf;

```

```

1283 }
1284 ;
1285
1286 struct rpc_msg
1287 {
1288     unsigned long rm_xid;
1289     enum msg_type rm_direction;
1290     union
1291     {
1292         struct call_body RM_cmb;
1293         struct reply_body RM_rmb;
1294     }
1295     ru;
1296 }
1297 ;

```

1.3.29. rpc/svc.h

```

1298
1299 #define svc_freeargs(xprt,xargs, argsp) (*(xprt)->xp_ops->xp_freeargs)((xprt), (xargs),
1300 (argsp))
1301 #define svc_getargs(xprt,xargs, argsp) (*(xprt)->xp_ops->xp_getargs)((xprt), (xargs),
1302 (argsp))
1303 #define RPC_ANYSOCK      -1
1304
1305 typedef struct SVCXPRT
1306 {
1307     int xp_sock;
1308     u_short xp_port;
1309     struct xp_ops *xp_ops;
1310     int xp_addrlen;
1311     struct sockaddr_in xp_raddr;
1312     struct opaque_auth xp_verf;
1313     caddr_t xp_p1;
1314     caddr_t xp_p2;
1315     char xp_pad[256];
1316 }
1317 SVCXPRT;
1318
1319 struct svc_req
1320 {
1321     rpcprog_t rq_prog;
1322     rpcvers_t rq_vers;
1323     rpcproc_t rq_proc;
1324     struct opaque_auth rq_cred;
1325     caddr_t rq_clntcred;
1326     SVCXPRT *rq_xprt;
1327 }
1328 ;
1329
1330 typedef void (*__dispatch_fn_t) (struct svc_req *, SVCXPRT *);
1331

```



```

1332 struct xp_ops
1333 {
1334     bool_t (*xp_recv) (SVCXPRT * __xpirt, struct rpc_msg * __msg);
1335     enum xpirt_stat (*xp_stat) (SVCXPRT * __xpirt);
1336     bool_t (*xp_getargs) (SVCXPRT * __xpirt, xdrproc_t __xdr_args,
1337                           caddr_t args_ptr);
1338     bool_t (*xp_reply) (SVCXPRT * __xpirt, struct rpc_msg * __msg);
1339     bool_t (*xp_freeargs) (SVCXPRT * __xpirt, xdrproc_t __xdr_args,
1340                           caddr_t args_ptr);
1341     void (*xp_destroy) (SVCXPRT * __xpirt);
1342 }
1343 ;

```

1.3.30. rpc/types.h

```

1344
1345 typedef int bool_t;
1346 typedef int enum_t;
1347 typedef unsigned long rpcprog_t;
1348 typedef unsigned long rpcvers_t;
1349 typedef unsigned long rpcproc_t;
1350 typedef unsigned long rpcprot_t;

```

1.3.31. rpc/xdr.h

```

1351
1352 enum xdr_op
1353 {
1354     XDR_ENCODE, XDR_DECODE, XDR_FREE
1355 }
1356 ;
1357 typedef struct XDR
1358 {
1359     enum xdr_op x_op;
1360     struct xdr_ops *x_ops;
1361     caddr_t x_public;
1362     caddr_t x_private;
1363     caddr_t x_base;
1364     int x_handy;
1365 }
1366 XDR;
1367
1368 struct xdr_ops
1369 {
1370     bool_t (*x_getlong) (XDR * __xdrs, long * __lp);
1371     bool_t (*x_putlong) (XDR * __xdrs, long * __lp);
1372     bool_t (*x_getbytes) (XDR * __xdrs, caddr_t __addr, u_int __len);
1373     bool_t (*x_putbytes) (XDR * __xdrs, char * __addr, u_int __len);
1374     u_int (*x_getpostn) (XDR * __xdrs);
1375     bool_t (*x_setpostn) (XDR * __xdrs, u_int __pos);
1376     int32_t (*x_inline) (XDR * __xdrs, int __len);
1377     void (*x_destroy) (XDR * __xdrs);

```

```

1378     bool_t (*x_getint32) (XDR * __xdrs, int32_t * __ip);
1379     bool_t (*x_putint32) (XDR * __xdrs, int32_t * __ip);
1380 }
1381 ;
1382
1383 typedef bool_t (*xdrproc_t) (XDR *, void *, ...);
1384
1385 struct xdr_discrim
1386 {
1387     int value;
1388     xdrproc_t proc;
1389 }
1390 ;

```

1.3.32. sched.h

```

1391
1392 #define SCHED_OTHER      0
1393 #define SCHED_FIFO      1
1394 #define SCHED_RR        2
1395
1396 struct sched_param
1397 {
1398     int sched_priority;
1399 }
1400 ;

```

1.3.33. search.h

```

1401
1402 typedef struct entry
1403 {
1404     char *key;
1405     void *data;
1406 }
1407 ENTRY;
1408 typedef enum
1409 {
1410     FIND, ENTER
1411 }
1412 ACTION;
1413 typedef enum
1414 {
1415     preorder, postorder, endorder, leaf
1416 }
1417 VISIT;
1418
1419 typedef void (*__action_fn_t) (void *__nodep, VISIT __value, int __level);

```

1.3.34. setjmp.h

```

1420

```

```

1421 #define setjmp(env)      _setjmp(env)
1422 #define sigsetjmp(a,b)  __sigsetjmp(a,b)
1423
1424 struct __jmp_buf_tag
1425 {
1426     __jmp_buf __jmpbuf;
1427     int __mask_was_saved;
1428     sigset_t __saved_mask;
1429 }
1430 ;
1431
1432 typedef struct __jmp_buf_tag jmp_buf[1];
1433 typedef jmp_buf sigjmp_buf;

```

1.3.35. signal.h

```

1434
1435 #define SIGRTMAX          (__libc_current_sigrtmax ())
1436 #define SIGRTMIN          (__libc_current_sigrtmin ())
1437 #define SIG_BLOCK         0
1438 #define SIG_UNBLOCK       1
1439 #define SIG_SETMASK       2
1440 #define NSIG              65
1441
1442 typedef int sig_atomic_t;
1443 struct sigstack
1444 {
1445     void *ss_sp;
1446     int ss_onstack;
1447 }
1448 ;
1449
1450 typedef void (*sighandler_t) (int);
1451 #define SIG_HOLD          ((sighandler_t) 2)
1452 #define SIG_ERR            ((sighandler_t)-1)
1453 #define SIG_DFL            ((sighandler_t)0)
1454 #define SIG_IGN            ((sighandler_t)1)
1455
1456 #define SIGHUP            1
1457 #define SIGUSR1           10
1458 #define SIGSEGV           11
1459 #define SIGUSR2           12
1460 #define SIGPIPE           13
1461 #define SIGALRM           14
1462 #define SIGTERM           15
1463 #define SIGSTKFLT         16
1464 #define SIGCHLD           17
1465 #define SIGCONT           18
1466 #define SIGSTOP           19
1467 #define SIGINT            2
1468 #define SIGTSTP           20
1469 #define SIGTTIN           21

```

```

1470 #define SIGTTOU 22
1471 #define SIGURG 23
1472 #define SIGXCPU 24
1473 #define SIGXFSZ 25
1474 #define SIGVTALRM 26
1475 #define SIGPROF 27
1476 #define SIGWINCH 28
1477 #define SIGIO 29
1478 #define SIGQUIT 3
1479 #define SIGPWR 30
1480 #define SIGSYS 31
1481 #define SIGUNUSED 31
1482 #define SIGILL 4
1483 #define SIGTRAP 5
1484 #define SIGABRT 6
1485 #define SIGIOT 6
1486 #define SIGBUS 7
1487 #define SIGFPE 8
1488 #define SIGKILL 9
1489 #define SIGCLD SIGCHLD
1490 #define SIGPOLL SIGIO
1491
1492 #define SV_ONSTACK (1<<0)
1493 #define SV_INTERRUPT (1<<1)
1494 #define SV_RESETHAND (1<<2)
1495
1496 typedef union sigval
1497 {
1498     int sival_int;
1499     void *sival_ptr;
1500 }
1501 sigval_t;
1502 #define SIGEV_SIGNAL 0
1503 #define SIGEV_NONE 1
1504 #define SIGEV_THREAD 2
1505
1506 typedef struct sigevent
1507 {
1508     sigval_t sigev_value;
1509     int sigev_signo;
1510     int sigev_notify;
1511     union
1512     {
1513         int _pad[SIGEV_PAD_SIZE];
1514         struct
1515         {
1516             void (*sigev_thread_func) (sigval_t);
1517             void *_attribute;
1518         }
1519         _sigev_thread;
1520     }
1521     _sigev_un;
1522 }

```

```

1523  sigevent_t;
1524  #define si_pid  _sifields._kill._pid
1525  #define si_uid  _sifields._kill._uid
1526  #define si_value  _sifields._rt._sigval
1527  #define si_int  _sifields._rt._sigval.sival_int
1528  #define si_ptr  _sifields._rt._sigval.sival_ptr
1529  #define si_status  _sifields._sigchld._status
1530  #define si_stime  _sifields._sigchld._stime
1531  #define si_utime  _sifields._sigchld._utime
1532  #define si_addr  _sifields._sigfault._addr
1533  #define si_band  _sifields._sigpoll._band
1534  #define si_fd  _sifields._sigpoll._fd
1535  #define si_timer1  _sifields._timer._timer1
1536  #define si_timer2  _sifields._timer._timer2
1537
1538  typedef struct siginfo
1539  {
1540      int si_signo;
1541      int si_errno;
1542      int si_code;
1543      union
1544      {
1545          int _pad[SI_PAD_SIZE];
1546          struct
1547          {
1548              pid_t _pid;
1549              uid_t _uid;
1550          }
1551          _kill;
1552          struct
1553          {
1554              unsigned int _timer1;
1555              unsigned int _timer2;
1556          }
1557          _timer;
1558          struct
1559          {
1560              pid_t _pid;
1561              uid_t _uid;
1562              sigval_t _sigval;
1563          }
1564          _rt;
1565          struct
1566          {
1567              pid_t _pid;
1568              uid_t _uid;
1569              int _status;
1570              clock_t _utime;
1571              clock_t _stime;
1572          }
1573          _sigchld;
1574          struct
1575          {

```

```

1576     void *_addr;
1577 }
1578 _sigfault;
1579 struct
1580 {
1581     int _band;
1582     int _fd;
1583 }
1584 _sigpoll;
1585 }
1586 _sifields;
1587 }
1588 siginfo_t;
1589 #define SI_QUEUE -1
1590 #define SI_TIMER -2
1591 #define SI_MESGQ -3
1592 #define SI_ASYNCIO -4
1593 #define SI_SIGIO -5
1594 #define SI_TKILL -6
1595 #define SI_ASYNCNL -60
1596 #define SI_USER 0
1597 #define SI_KERNEL 0x80
1598
1599 #define ILL_ILLOPC 1
1600 #define ILL_ILLOPN 2
1601 #define ILL_ILLADR 3
1602 #define ILL_ILLTRP 4
1603 #define ILL_PRVOPC 5
1604 #define ILL_PRVREG 6
1605 #define ILL_COPROC 7
1606 #define ILL_BADSTK 8
1607
1608 #define FPE_INTDIV 1
1609 #define FPE_INTOVF 2
1610 #define FPE_FLTDIV 3
1611 #define FPE_FLTOVF 4
1612 #define FPE_FLTUND 5
1613 #define FPE_FLTRES 6
1614 #define FPE_FLTINV 7
1615 #define FPE_FLTSUB 8
1616
1617 #define SEGV_MAPERR 1
1618 #define SEGV_ACCERR 2
1619
1620 #define BUS_ADRALN 1
1621 #define BUS_ADRERR 2
1622 #define BUS_OBJERR 3
1623
1624 #define TRAP_BRKPT 1
1625 #define TRAP_TRACE 2
1626
1627 #define CLD_EXITED 1
1628 #define CLD_KILLED 2

```

```

1629 #define CLD_DUMPED      3
1630 #define CLD_TRAPPED     4
1631 #define CLD_STOPPED     5
1632 #define CLD_CONTINUED   6
1633
1634 #define POLL_IN 1
1635 #define POLL_OUT      2
1636 #define POLL_MSG      3
1637 #define POLL_ERR      4
1638 #define POLL_PRI      5
1639 #define POLL_HUP      6
1640
1641 typedef struct
1642 {
1643     unsigned long sig[_SIGSET_NWORDS];
1644 }
1645 sigset_t;
1646 #define SA_NOCLDSTOP    0x00000001
1647 #define SA_NOCLDWAIT    0x00000002
1648 #define SA_SIGINFO      0x00000004
1649 #define SA_ONSTACK      0x08000000
1650 #define SA_RESTART      0x10000000
1651 #define SA_INTERRUPT     0x20000000
1652 #define SA_NODEFER       0x40000000
1653 #define SA_RESETHAND     0x80000000
1654 #define SA_NOMASK        SA_NODEFER
1655 #define SA_ONESHOT       SA_RESETHAND
1656
1657 typedef struct sigaltstack
1658 {
1659     void *ss_sp;
1660     int ss_flags;
1661     size_t ss_size;
1662 }
1663 stack_t;
1664 #define SS_ONSTACK      1
1665 #define SS_DISABLE      2

```

1.3.36. stddef.h

```

1666
1667 #define offsetof(TYPE, MEMBER) ((size_t)& ((TYPE*)0)->MEMBER)
1668 #define NULL      (0L)
1669
1670 typedef int wchar_t;

```

1.3.37. stdio.h

```

1671
1672 #define EOF      (-1)
1673 #define P_tmpdir  "/"tmp"
1674 #define FOPEN_MAX  16

```

```

1675 #define L_tmpnam      20
1676 #define FILENAME_MAX  4096
1677 #define BUFSIZ      8192
1678 #define L_ctermid    9
1679 #define L_cuserid     9
1680
1681 typedef struct
1682 {
1683     off_t __pos;
1684     mbstate_t __state;
1685 }
1686 fpos_t;
1687 typedef struct
1688 {
1689     off64_t __pos;
1690     mbstate_t __state;
1691 }
1692 fpos64_t;
1693
1694 typedef struct _IO_FILE FILE;
1695 #define _IOFBF 0
1696 #define _IOLBF 1
1697 #define _IONBF 2

```

1.3.38. stdlib.h

```

1698
1699 #define MB_CUR_MAX      (__ctype_get_mb_cur_max())
1700 #define EXIT_SUCCESS    0
1701 #define EXIT_FAILURE    1
1702 #define RAND_MAX        2147483647
1703
1704 typedef int (*__compar_fn_t) (const void *, const void *);
1705 struct random_data
1706 {
1707     int32_t *fptr;
1708     int32_t *rptr;
1709     int32_t *state;
1710     int rand_type;
1711     int rand_deg;
1712     int rand_sep;
1713     int32_t *end_ptr;
1714 }
1715 ;
1716
1717 typedef struct
1718 {
1719     int quot;
1720     int rem;
1721 }
1722 div_t;
1723

```



```

1724 typedef struct
1725 {
1726     long quot;
1727     long rem;
1728 }
1729 ldiv_t;
1730
1731 typedef struct
1732 {
1733     long long quot;
1734     long long rem;
1735 }
1736 lldiv_t;

```

1.3.39. sys/file.h

```

1737
1738 #define LOCK_SH 1
1739 #define LOCK_EX 2
1740 #define LOCK_NB 4
1741 #define LOCK_UN 8

```

1.3.40. sys/ipc.h

```

1742
1743 #define IPC_PRIVATE ((key_t)0)
1744 #define IPC_RMID 0
1745 #define IPC_CREAT 00001000
1746 #define IPC_EXCL 00002000
1747 #define IPC_NOWAIT 00004000
1748 #define IPC_SET 1
1749 #define IPC_STAT 2

```

1.3.41. sys/mman.h

```

1750
1751 #define MAP_FAILED ((void*)-1)
1752 #define PROT_NONE 0x0
1753 #define MAP_SHARED 0x01
1754 #define MAP_PRIVATE 0x02
1755 #define PROT_READ 0x1
1756 #define MAP_FIXED 0x10
1757 #define PROT_WRITE 0x2
1758 #define MAP_ANONYMOUS 0x20
1759 #define PROT_EXEC 0x4
1760 #define MS_ASYNC 1
1761 #define MS_INVALIDATE 2
1762 #define MS_SYNC 4
1763 #define MAP_ANON MAP_ANONYMOUS

```

1.3.42. sys/msg.h

```

1764
1765 #define MSG_NOERROR      010000

```

1.3.43. sys/param.h

```

1766
1767 #define NOFILE    256
1768 #define MAXPATHLEN 4096

```

1.3.44. sys/poll.h

```

1769
1770 #define POLLIN    0x0001
1771 #define POLLPRI   0x0002
1772 #define POLLOUT   0x0004
1773 #define POLLERR   0x0008
1774 #define POLLHUP   0x0010
1775 #define POLLNVAL  0x0020
1776
1777 struct pollfd
1778 {
1779     int fd;
1780     short events;
1781     short revents;
1782 }
1783 ;
1784 typedef unsigned long nfd_t;

```

1.3.45. sys/resource.h

```

1785
1786 #define RUSAGE_CHILDREN (-1)
1787 #define RUSAGE_BOTH     (-2)
1788 #define RLIM_INFINITY  (~0UL)
1789 #define RLIM_SAVED_CUR  -1
1790 #define RLIM_SAVED_MAX  -1
1791 #define RLIMIT_CPU      0
1792 #define RUSAGE_SELF     0
1793 #define RLIMIT_FSIZE    1
1794 #define RLIMIT_DATA     2
1795 #define RLIMIT_STACK    3
1796 #define RLIMIT_CORE     4
1797 #define RLIMIT_NOFILE   7
1798 #define RLIMIT_AS       9
1799
1800 typedef unsigned long rlim_t;
1801 typedef unsigned long long rlim64_t;
1802 typedef int __rlimit_resource_t;
1803

```

```

1804 struct rlimit
1805 {
1806     rlim_t rlim_cur;
1807     rlim_t rlim_max;
1808 }
1809 ;
1810 struct rlimit64
1811 {
1812     rlim64_t rlim_cur;
1813     rlim64_t rlim_max;
1814 }
1815 ;
1816
1817 struct rusage
1818 {
1819     struct timeval ru_utime;
1820     struct timeval ru_stime;
1821     long ru_maxrss;
1822     long ru_ixrss;
1823     long ru_idrss;
1824     long ru_isrss;
1825     long ru_minflt;
1826     long ru_majflt;
1827     long ru_nswap;
1828     long ru_inblock;
1829     long ru_oublock;
1830     long ru_msgsnd;
1831     long ru_msgrcv;
1832     long ru_nsignals;
1833     long ru_nvcsw;
1834     long ru_nivcsw;
1835 }
1836 ;
1837
1838 enum __priority_which
1839 {
1840     PRIO_PROCESS, PRIO_PGRP = 1, PRIO_USER = 2
1841 }
1842 ;
1843 #define PRIO_PGRP          PRIO_PGRP
1844 #define PRIO_PROCESS       PRIO_PROCESS
1845 #define PRIO_USER          PRIO_USER
1846
1847 typedef enum __priority_which __priority_which_t;

```

1.3.46. sys/sem.h

```

1848
1849 #define SEM_UNDO          0x1000
1850 #define GETPID    11
1851 #define GETVAL    12
1852 #define GETALL    13

```

```

1853 #define GETNCNT 14
1854 #define GETZCNT 15
1855 #define SETVAL 16
1856 #define SETALL 17
1857
1858 struct sembuf
1859 {
1860     short sem_num;
1861     short sem_op;
1862     short sem_flg;
1863 }
1864 ;

```

1.3.47. sys/shm.h

```

1865
1866 #define SHM_RDONLY      010000
1867 #define SHM_W           0200
1868 #define SHM_RND 020000
1869 #define SHM_R           0400
1870 #define SHM_REMAP       040000
1871 #define SHM_LOCK        11
1872 #define SHM_UNLOCK      12

```

1.3.48. sys/socket.h

```

1873
1874 #define SHUT_RD 0
1875 #define MSG_WAITALL      0x100
1876 #define MSG_TRUNC        0x20
1877 #define MSG_EOR 0x80
1878 #define SIOCGIFCONF      0x8912
1879 #define SIOCGIFFLAGS     0x8913
1880 #define SIOCGIFADDR      0x8915
1881 #define SIOCGIFNETMASK   0x891b
1882 #define MSG_OOB 1
1883 #define SHUT_WR 1
1884 #define MSG_PEEK          2
1885 #define SHUT_RDWR        2
1886 #define MSG_DONTROUTE    4
1887 #define MSG_CTRUNC        8
1888 #define PF_UNSPEC        AF_UNSPEC
1889
1890 struct linger
1891 {
1892     int l_onoff;
1893     int l_linger;
1894 }
1895 ;
1896 struct cmsghdr
1897 {
1898     size_t cmsg_len;

```

```

1899     int cmsg_level;
1900     int cmsg_type;
1901 }
1902 ;
1903 struct iovec
1904 {
1905     void *iov_base;
1906     size_t iov_len;
1907 }
1908 ;
1909
1910 typedef unsigned short sa_family_t;
1911 typedef unsigned int socklen_t;
1912
1913 struct sockaddr
1914 {
1915     sa_family_t sa_family;
1916     char sa_data[14];
1917 }
1918 ;
1919 struct sockaddr_storage
1920 {
1921     sa_family_t ss_family;
1922     __ss_aligntype __ss_align;
1923     char __ss_padding[(128 - (2 * sizeof (__ss_aligntype)))];
1924 }
1925 ;
1926
1927 struct msghdr
1928 {
1929     void *msg_name;
1930     int msg_namelen;
1931     struct iovec *msg_iov;
1932     size_t msg_iovlen;
1933     void *msg_control;
1934     size_t msg_controllen;
1935     unsigned int msg_flags;
1936 }
1937 ;
1938 #define AF_UNSPEC      0
1939 #define AF_UNIX 1
1940 #define AF_INET6      10
1941 #define AF_INET 2
1942
1943 #define PF_INET AF_INET
1944 #define PF_INET6 AF_INET6
1945 #define PF_UNIX AF_UNIX
1946
1947 #define SOCK_STREAM    1
1948 #define SOCK_PACKET    10
1949 #define SOCK_DGRAM     2
1950 #define SOCK_RAW       3
1951 #define SOCK_RDM       4

```

```

1952 #define SOCK_SEQPACKET 5
1953
1954 #define SOL_SOCKET 1
1955 #define SO_DEBUG 1
1956 #define SO_OOBINLINE 10
1957 #define SO_NO_CHECK 11
1958 #define SO_PRIORITY 12
1959 #define SO_LINGER 13
1960 #define SO_REUSEADDR 2
1961 #define SOL_RAW 255
1962 #define SO_TYPE 3
1963 #define SO_ERROR 4
1964 #define SO_DONTROUTE 5
1965 #define SO_BROADCAST 6
1966 #define SO_SNDBUF 7
1967 #define SO_RCVBUF 8
1968 #define SO_KEEPALIVE 9

```

1.3.49. sys/stat.h

```

1969
1970 #define S_ISBLK(m) ((m) & S_IFMT) == S_IFBLK)
1971 #define S_ISCHR(m) ((m) & S_IFMT) == S_IFCHR)
1972 #define S_ISDIR(m) ((m) & S_IFMT) == S_IFDIR)
1973 #define S_ISFIFO(m) ((m) & S_IFMT) == S_IFIFO)
1974 #define S_ISLNK(m) ((m) & S_IFMT) == S_IFLNK)
1975 #define S_ISREG(m) ((m) & S_IFMT) == S_IFREG)
1976 #define S_ISSOCK(m) ((m) & S_IFMT) == S_IFSOCK)
1977 #define S_TYPEISMQ(buf) ((buf)->st_mode - (buf)->st_mode)
1978 #define S_TYPEISSEM(buf) ((buf)->st_mode - (buf)->st_mode)
1979 #define S_TYPEISSHM(buf) ((buf)->st_mode - (buf)->st_mode)
1980 #define S_IRWXU (S_IREAD|S_IWRITE|S_IEXEC)
1981 #define S_IROTH (S_IRGRP>>3)
1982 #define S_IRGRP (S_IRUSR>>3)
1983 #define S_IRWXO (S_IRWXG>>3)
1984 #define S_IRWXG (S_IRWXU>>3)
1985 #define S_IWOTH (S_IWGRP>>3)
1986 #define S_IWGRP (S_IWUSR>>3)
1987 #define S_IXOTH (S_IXGRP>>3)
1988 #define S_IXGRP (S_IXUSR>>3)
1989 #define S_ISVTX 01000
1990 #define S_IXUSR 0x0040
1991 #define S_IWUSR 0x0080
1992 #define S_IRUSR 0x0100
1993 #define S_ISGID 0x0400
1994 #define S_ISUID 0x0800
1995 #define S_IFIFO 0x1000
1996 #define S_IFCHR 0x2000
1997 #define S_IFDIR 0x4000
1998 #define S_IFBLK 0x6000
1999 #define S_IFREG 0x8000
2000 #define S_IFLNK 0xa000

```

```

2001 #define S_IFSOCK      0xc000
2002 #define S_IFMT  0xf000
2003 #define st_atime      st_atim.tv_sec
2004 #define st_ctime      st_ctim.tv_sec
2005 #define st_mtime      st_mtim.tv_sec
2006 #define S_IREAD S_IRUSR
2007 #define S_IWRITE S_IWUSR
2008 #define S_IEXEC S_IXUSR

```

1.3.50. sys/time.h

```

2009
2010 #define ITIMER_REAL      0
2011 #define ITIMER_VIRTUAL  1
2012 #define ITIMER_PROF      2
2013
2014 struct timezone
2015 {
2016     int tz_minuteswest;
2017     int tz_dsttime;
2018 }
2019 ;
2020
2021 typedef int __itimer_which_t;
2022
2023 struct timespec
2024 {
2025     time_t tv_sec;
2026     long tv_nsec;
2027 }
2028 ;
2029
2030 struct timeval
2031 {
2032     time_t tv_sec;
2033     suseconds_t tv_usec;
2034 }
2035 ;
2036
2037 struct itimerval
2038 {
2039     struct timeval it_interval;
2040     struct timeval it_value;
2041 }
2042 ;

```

1.3.51. sys/timeb.h

```

2043
2044 struct timeb
2045 {
2046     time_t time;

```

```

2047     unsigned short millitm;
2048     short timezone;
2049     short dstflag;
2050 }
2051 ;

```

1.3.52. sys/times.h

```

2052
2053 struct tms
2054 {
2055     clock_t tms_utime;
2056     clock_t tms_stime;
2057     clock_t tms_cutime;
2058     clock_t tms_cstime;
2059 }
2060 ;

```

1.3.53. sys/types.h

```

2061
2062 #define FD_ISSET(d,set) ((set)->fds_bits[((d)/(8*sizeof(long)))]&
2063 (1<<((d)%(8*sizeof(long)))))
2064 #define FD_CLR(d,set) ((set)->fds_bits[((d)/(8*sizeof(long)))]&
2065 ~(1<<((d)%(8*sizeof(long)))))
2066 #define FD_SET(d,set)
2067 ((set)->fds_bits[((d)/(8*sizeof(long)))]|=(1<<((d)%(8*sizeof(long)))))
2068 #define FALSE 0
2069 #define TRUE 1
2070 #define FD_SETSIZE 1024
2071 #define FD_ZERO(fdsetp) bzero(fdsetp, sizeof(*(fdsetp)))
2072
2073 typedef signed char int8_t;
2074 typedef short int16_t;
2075 typedef int int32_t;
2076 typedef unsigned char u_int8_t;
2077 typedef unsigned short u_int16_t;
2078 typedef unsigned int u_int32_t;
2079 typedef unsigned int uid_t;
2080 typedef int pid_t;
2081 typedef unsigned long off_t;
2082 typedef int key_t;
2083 typedef long suseconds_t;
2084 typedef unsigned int u_int;
2085 typedef struct
2086 {
2087     int __val[2];
2088 }
2089 fsid_t;
2090 typedef unsigned int useconds_t;
2091 typedef unsigned long blksize_t;
2092 typedef long fd_mask;

```



```

2093 typedef int timer_t;
2094 typedef int clockid_t;
2095
2096 typedef unsigned int id_t;
2097
2098 typedef unsigned long long ino64_t;
2099 typedef long long loff_t;
2100 typedef unsigned long blkcnt_t;
2101 typedef unsigned long fsblkcnt_t;
2102 typedef unsigned long fsfilcnt_t;
2103 typedef unsigned long long blkcnt64_t;
2104 typedef unsigned long long fsblkcnt64_t;
2105 typedef unsigned long long fsfilcnt64_t;
2106 typedef unsigned char u_char;
2107 typedef unsigned short u_short;
2108 typedef unsigned long u_long;
2109
2110 typedef unsigned long ino_t;
2111 typedef unsigned int gid_t;
2112 typedef unsigned long long dev_t;
2113 typedef unsigned int mode_t;
2114 typedef unsigned long nlink_t;
2115 typedef char *caddr_t;
2116
2117 typedef struct
2118 {
2119     unsigned long fds_bits[__FDSET_LONGS];
2120 }
2121 fd_set;
2122
2123 typedef long clock_t;
2124 typedef long time_t;

```

1.3.54. sys/un.h

```

2125
2126 #define UNIX_PATH_MAX    108
2127
2128 struct sockaddr_un
2129 {
2130     sa_family_t sun_family;
2131     char sun_path[UNIX_PATH_MAX];
2132 }
2133 ;

```

1.3.55. sys/utsname.h

```

2134
2135 #define SYS_NMLN          65
2136
2137 struct utsname
2138 {

```

```

2139     char sysname[65];
2140     char nodename[65];
2141     char release[65];
2142     char version[65];
2143     char machine[65];
2144     char domainname[65];
2145 }
2146 ;

```

1.3.56. sys/wait.h

```

2147
2148 #define WIFSIGNALED(status)      (!WIFSTOPPED(status) & & !WIFEXITED(status))
2149 #define WIFSTOPPED(status)      (((status) & 0xff) == 0x7f)
2150 #define WEXITSTATUS(status)     (((status) & 0xff00) >> 8)
2151 #define WTERMSIG(status)        ((status) & 0x7f)
2152 #define WCOREDUMP(status)       ((status) & 0x80)
2153 #define WIFEXITED(status)       (WTERMSIG(status) == 0)
2154 #define WNOHANG 0x00000001
2155 #define WUNTRACED 0x00000002
2156 #define WCOREFLAG 0x80
2157 #define WSTOPSIG(status)        WEXITSTATUS(status)
2158
2159 typedef enum
2160 {
2161     P_ALL, P_PID, P_PGID
2162 }
2163 idtype_t;

```

1.3.57. syslog.h

```

2164
2165 #define LOG_EMERG 0
2166 #define LOG_PRIMASK 0x07
2167 #define LOG_ALERT 1
2168 #define LOG_CRIT 2
2169 #define LOG_ERR 3
2170 #define LOG_WARNING 4
2171 #define LOG_NOTICE 5
2172 #define LOG_INFO 6
2173 #define LOG_DEBUG 7
2174
2175 #define LOG_KERN (0<<3)
2176 #define LOG_AUTHPRIV (10<<3)
2177 #define LOG_FTP (11<<3)
2178 #define LOG_USER (1<<3)
2179 #define LOG_MAIL (2<<3)
2180 #define LOG_DAEMON (3<<3)
2181 #define LOG_AUTH (4<<3)
2182 #define LOG_SYSLOG (5<<3)
2183 #define LOG_LPR (6<<3)
2184 #define LOG_NEWS (7<<3)

```

```

2185 #define LOG_UUCP          (8<<3)
2186 #define LOG_CRON          (9<<3)
2187 #define LOG_FACMASK      0x03f8
2188
2189 #define LOG_LOCAL0        (16<<3)
2190 #define LOG_LOCAL1        (17<<3)
2191 #define LOG_LOCAL2        (18<<3)
2192 #define LOG_LOCAL3        (19<<3)
2193 #define LOG_LOCAL4        (20<<3)
2194 #define LOG_LOCAL5        (21<<3)
2195 #define LOG_LOCAL6        (22<<3)
2196 #define LOG_LOCAL7        (23<<3)
2197
2198 #define LOG_UPTO(pri)     ((1 << ((pri)+1)) - 1)
2199 #define LOG_MASK(pri)     (1 << (pri))
2200
2201 #define LOG_PID 0x01
2202 #define LOG_CONS      0x02
2203 #define LOG_ODELAY     0x04
2204 #define LOG_NDELAY     0x08
2205 #define LOG_NOWAIT     0x10
2206 #define LOG_PERROR     0x20

```

1.3.58. termios.h

```

2207
2208 #define TCIFLUSH          0
2209 #define TCOOFF           0
2210 #define TCSANOW          0
2211 #define BS0              0000000
2212 #define CR0              0000000
2213 #define FF0              0000000
2214 #define NL0              0000000
2215 #define TAB0             0000000
2216 #define VT0              0000000
2217 #define OPOST            0000001
2218 #define OCRNL            0000010
2219 #define ONOCR            0000020
2220 #define ONLRET           0000040
2221 #define OFILL            0000100
2222 #define OFDEL            0000200
2223 #define NL1              0000400
2224 #define TCOFLUSH         1
2225 #define TCOON            1
2226 #define TCSADRAIN        1
2227 #define TCIOFF           2
2228 #define TCIOFLUSH        2
2229 #define TCSAFLUSH        2
2230 #define TCION            3
2231
2232 typedef unsigned int speed_t;
2233 typedef unsigned char cc_t;

```

```

2234 typedef unsigned int tcflag_t;
2235 #define NCCS      32
2236
2237 struct termios
2238 {
2239     tcflag_t c_iflag;
2240     tcflag_t c_oflag;
2241     tcflag_t c_cflag;
2242     tcflag_t c_lflag;
2243     cc_t c_line;
2244     cc_t c_cc[NCCS];
2245     speed_t c_ispeed;
2246     speed_t c_ospeed;
2247 }
2248 ;
2249 #define VINTR      0
2250 #define VQUIT      1
2251 #define VLNEXT     15
2252 #define VERASE      2
2253 #define VKILL       3
2254 #define VEOF        4
2255
2256 #define IGNBRK     0000001
2257 #define BRKINT     0000002
2258 #define IGNPAR     0000004
2259 #define PARMRK     0000010
2260 #define INPCK      0000020
2261 #define ISTRIP     0000040
2262 #define INLCR      0000100
2263 #define IGNCR      0000200
2264 #define ICRNL      0000400
2265 #define IXANY      0004000
2266 #define IMAXBEL    0020000
2267
2268 #define CS5         0000000
2269
2270 #define ECHO        0000010
2271
2272 #define B0          0000000
2273 #define B50         0000001
2274 #define B75         0000002
2275 #define B110        0000003
2276 #define B134        0000004
2277 #define B150        0000005
2278 #define B200        0000006
2279 #define B300        0000007
2280 #define B600        0000010
2281 #define B1200       0000011
2282 #define B1800       0000012
2283 #define B2400       0000013
2284 #define B4800       0000014
2285 #define B9600       0000015
2286 #define B19200      0000016

```

```
2287 #define B38400 0000017
```

1.3.59. time.h

```
2288
2289 #define CLK_TCK ((clock_t)__sysconf(2))
2290 #define CLOCK_REALTIME 0
2291 #define TIMER_ABSTIME 1
2292 #define CLOCKS_PER_SEC 1000000L
2293
2294 struct tm
2295 {
2296     int tm_sec;
2297     int tm_min;
2298     int tm_hour;
2299     int tm_mday;
2300     int tm_mon;
2301     int tm_year;
2302     int tm_wday;
2303     int tm_yday;
2304     int tm_isdst;
2305     long tm_gmtoff;
2306     char *tm_zone;
2307 }
2308 ;
2309 struct itimerspec
2310 {
2311     struct timespec it_interval;
2312     struct timespec it_value;
2313 }
2314 ;
```

1.3.60. ulimit.h

```
2315
2316 #define UL_GETFSIZE 1
2317 #define UL_SETFSIZE 2
```

1.3.61. unistd.h

```
2318
2319 #define SEEK_SET 0
2320 #define STDIN_FILENO 0
2321 #define SEEK_CUR 1
2322 #define STDOUT_FILENO 1
2323 #define SEEK_END 2
2324 #define STDERR_FILENO 2
2325
2326 typedef long long off64_t;
2327 #define F_OK 0
2328 #define X_OK 1
2329 #define W_OK 2
```

```

2330 #define R_OK      4
2331
2332 #define _POSIX_VDISABLE '\0'
2333 #define _POSIX_CHOWN_RESTRICTED 1
2334 #define _POSIX_JOB_CONTROL      1
2335 #define _POSIX_NO_TRUNC         1
2336 #define _POSIX_SHELL            1
2337 #define _POSIX_FSYNC            200112
2338 #define _POSIX_MAPPED_FILES     200112
2339 #define _POSIX_MEMLOCK          200112
2340 #define _POSIX_MEMLOCK_RANGE    200112
2341 #define _POSIX_MEMORY_PROTECTION 200112
2342 #define _POSIX_SEMAPHORES       200112
2343 #define _POSIX_SHARED_MEMORY_OBJECTS 200112
2344 #define _POSIX_TIMERS           200112
2345 #define _POSIX2_C_BIND          200112L
2346 #define _POSIX2_VERSION         200112L
2347 #define _POSIX_THREADS          200112L
2348 #define _POSIX_VERSION          200112L
2349
2350 #define _PC_LINK_MAX            0
2351 #define _PC_MAX_CANON           1
2352 #define _PC_ASYNC_IO            10
2353 #define _PC_PRIO_IO             11
2354 #define _PC_FILESIZEBITS        13
2355 #define _PC_REC_INCR_XFER_SIZE  14
2356 #define _PC_REC_MIN_XFER_SIZE   16
2357 #define _PC_REC_XFER_ALIGN      17
2358 #define _PC_ALLOC_SIZE_MIN      18
2359 #define _PC_MAX_INPUT           2
2360 #define _PC_2_SYMLINKS          20
2361 #define _PC_NAME_MAX            3
2362 #define _PC_PATH_MAX            4
2363 #define _PC_PIPE_BUF            5
2364 #define _PC_CHOWN_RESTRICTED    6
2365 #define _PC_NO_TRUNC            7
2366 #define _PC_VDISABLE            8
2367 #define _PC_SYNC_IO             9
2368
2369 #define _SC_ARG_MAX             0
2370 #define _SC_CHILD_MAX           1
2371 #define _SC_PRIORITY_SCHEDULING 10
2372 #define _SC_TIMERS              11
2373 #define _SC_ASYNCHRONOUS_IO     12
2374 #define _SC_XBS5_ILP32_OFF32    125
2375 #define _SC_XBS5_ILP32_OFFBIG   126
2376 #define _SC_XBS5_LP64_OFF64     127
2377 #define _SC_XBS5_LPBIG_OFFBIG   128
2378 #define _SC_XOPEN_LEGACY        129
2379 #define _SC_PRIORITIZED_IO      13
2380 #define _SC_XOPEN_REALTIME      130
2381 #define _SC_XOPEN_REALTIME_THREADS 131
2382 #define _SC_ADVISORY_INFO       132

```

```

2383 #define _SC_BARRIERS      133
2384 #define _SC_CLOCK_SELECTION 137
2385 #define _SC_CPUTIME        138
2386 #define _SC_THREAD_CPUTIME 139
2387 #define _SC_SYNCHRONIZED_IO 14
2388 #define _SC_MONOTONIC_CLOCK 149
2389 #define _SC_FSYNC          15
2390 #define _SC_READER_WRITER_LOCKS 153
2391 #define _SC_SPIN_LOCKS     154
2392 #define _SC_REGEX          155
2393 #define _SC_SHELL          157
2394 #define _SC_SPAWN          159
2395 #define _SC_MAPPED_FILES   16
2396 #define _SC_SPORADIC_SERVER 160
2397 #define _SC_THREAD_SPORADIC_SERVER 161
2398 #define _SC_TIMEOUTS       164
2399 #define _SC_TYPED_MEMORY_OBJECTS 165
2400 #define _SC_2_PBS_ACCOUNTING 169
2401 #define _SC_MEMLOCK        17
2402 #define _SC_2_PBS_LOCATE   170
2403 #define _SC_2_PBS_MESSAGE  171
2404 #define _SC_2_PBS_TRACK    172
2405 #define _SC_SYMLINK_MAX    173
2406 #define _SC_2_PBS_CHECKPOINT 175
2407 #define _SC_V6_ILP32_OFF32 176
2408 #define _SC_V6_ILP32_OFFBIG 177
2409 #define _SC_V6_LP64_OFF64  178
2410 #define _SC_V6_LPBIG_OFFBIG 179
2411 #define _SC_MEMLOCK_RANGE  18
2412 #define _SC_HOST_NAME_MAX   180
2413 #define _SC_TRACE           181
2414 #define _SC_TRACE_EVENT_FILTER 182
2415 #define _SC_TRACE_INHERIT   183
2416 #define _SC_TRACE_LOG       184
2417 #define _SC_MEMORY_PROTECTION 19
2418 #define _SC_CLK_TCK         2
2419 #define _SC_MESSAGE_PASSING 20
2420 #define _SC_SEMAPHORES      21
2421 #define _SC_SHARED_MEMORY_OBJECTS 22
2422 #define _SC_AIO_LISTIO_MAX  23
2423 #define _SC_AIO_MAX          24
2424 #define _SC_AIO_PRIO_DELTA_MAX 25
2425 #define _SC_DELAYTIMER_MAX  26
2426 #define _SC_MQ_OPEN_MAX     27
2427 #define _SC_MQ_PRIO_MAX     28
2428 #define _SC_VERSION          29
2429 #define _SC_NGROUPS_MAX     3
2430 #define _SC_PAGESIZE        30
2431 #define _SC_PAGE_SIZE       30
2432 #define _SC_RTSIG_MAX        31
2433 #define _SC_SEM_NSEMS_MAX    32
2434 #define _SC_SEM_VALUE_MAX    33
2435 #define _SC_SIGQUEUE_MAX     34

```

```

2436 #define _SC_TIMER_MAX      35
2437 #define _SC_BC_BASE_MAX    36
2438 #define _SC_BC_DIM_MAX     37
2439 #define _SC_BC_SCALE_MAX    38
2440 #define _SC_BC_STRING_MAX   39
2441 #define _SC_OPEN_MAX        4
2442 #define _SC_COLL_WEIGHTS_MAX 40
2443 #define _SC_EXPR_NEST_MAX   42
2444 #define _SC_LINE_MAX        43
2445 #define _SC_RE_DUP_MAX      44
2446 #define _SC_2_VERSION        46
2447 #define _SC_2_C_BIND         47
2448 #define _SC_2_C_DEV          48
2449 #define _SC_2_FORT_DEV       49
2450 #define _SC_STREAM_MAX       5
2451 #define _SC_2_FORT_RUN       50
2452 #define _SC_2_SW_DEV         51
2453 #define _SC_2_LOCALEDEF      52
2454 #define _SC_TZNAME_MAX       6
2455 #define _SC_IOV_MAX          60
2456 #define _SC_THREADS          67
2457 #define _SC_THREAD_SAFE_FUNCTIONS 68
2458 #define _SC_GETGR_R_SIZE_MAX 69
2459 #define _SC_JOB_CONTROL      7
2460 #define _SC_GETPW_R_SIZE_MAX 70
2461 #define _SC_LOGIN_NAME_MAX   71
2462 #define _SC_TTY_NAME_MAX     72
2463 #define _SC_THREAD_DESTRUCTOR_ITERATIONS 73
2464 #define _SC_THREAD_KEYS_MAX  74
2465 #define _SC_THREAD_STACK_MIN 75
2466 #define _SC_THREAD_THREADS_MAX 76
2467 #define _SC_THREAD_ATTR_STACKADDR 77
2468 #define _SC_THREAD_ATTR_STACKSIZE 78
2469 #define _SC_THREAD_PRIORITY_SCHEDULING 79
2470 #define _SC_SAVED_IDS        8
2471 #define _SC_THREAD_PRIO_INHERIT 80
2472 #define _SC_THREAD_PRIO_PROTECT 81
2473 #define _SC_THREAD_PROCESS_SHARED 82
2474 #define _SC_ATEXIT_MAX       87
2475 #define _SC_PASS_MAX         88
2476 #define _SC_XOPEN_VERSION    89
2477 #define _SC_REALTIME_SIGNALS 9
2478 #define _SC_XOPEN_UNIX       91
2479 #define _SC_XOPEN_CRYPT      92
2480 #define _SC_XOPEN_ENH_I18N   93
2481 #define _SC_XOPEN_SHM        94
2482 #define _SC_2_CHAR_TERM      95
2483 #define _SC_2_C_VERSION      96
2484 #define _SC_2_UPE            97
2485
2486 #define _CS_PATH              0
2487 #define _POSIX_REGEX          1
2488 #define _CS_XBS5_ILP32_OFF32_CFLAGS 1100

```



```

2489 #define _CS_XBS5_ILP32_OFF32_LDFLAGS 1101
2490 #define _CS_XBS5_ILP32_OFF32_LIBS 1102
2491 #define _CS_XBS5_ILP32_OFF32_LINTFLAGS 1103
2492 #define _CS_XBS5_ILP32_OFFBIG_CFLAGS 1104
2493 #define _CS_XBS5_ILP32_OFFBIG_LDFLAGS 1105
2494 #define _CS_XBS5_ILP32_OFFBIG_LIBS 1106
2495 #define _CS_XBS5_ILP32_OFFBIG_LINTFLAGS 1107
2496 #define _CS_XBS5_LP64_OFF64_CFLAGS 1108
2497 #define _CS_XBS5_LP64_OFF64_LDFLAGS 1109
2498 #define _CS_XBS5_LP64_OFF64_LIBS 1110
2499 #define _CS_XBS5_LP64_OFF64_LINTFLAGS 1111
2500 #define _CS_XBS5_LPBIG_OFFBIG_CFLAGS 1112
2501 #define _CS_XBS5_LPBIG_OFFBIG_LDFLAGS 1113
2502 #define _CS_XBS5_LPBIG_OFFBIG_LIBS 1114
2503 #define _CS_XBS5_LPBIG_OFFBIG_LINTFLAGS 1115
2504
2505 #define _XOPEN_REALTIME 1
2506 #define _XOPEN_XPG4 1
2507 #define _XOPEN_XCU_VERSION 4
2508 #define _XOPEN_VERSION 500
2509
2510 #define F_ULOCK 0
2511 #define F_LOCK 1
2512 #define F_TLOCK 2
2513 #define F_TEST 3

```

1.3.62. utime.h

```

2514
2515 struct utimbuf
2516 {
2517     time_t actime;
2518     time_t modtime;
2519 }
2520 ;

```

1.3.63. utmp.h

```

2521
2522 #define UT_HOSTSIZE 256
2523 #define UT_LINESIZE 32
2524 #define UT_NAMESIZE 32
2525
2526 struct exit_status
2527 {
2528     short e_termination;
2529     short e_exit;
2530 }
2531 ;
2532
2533 #define EMPTY 0
2534 #define RUN_LVL 1

```

```

2535 #define BOOT_TIME      2
2536 #define NEW_TIME       3
2537 #define OLD_TIME       4
2538 #define INIT_PROCESS   5
2539 #define LOGIN_PROCESS   6
2540 #define USER_PROCESS   7
2541 #define DEAD_PROCESS    8
2542 #define ACCOUNTING     9

```

1.3.64. wchar.h

```

2543
2544 #define WEOF      (0xffffffffu)
2545 #define WCHAR_MAX 0x7FFFFFFF
2546 #define WCHAR_MIN 0x80000000

```

1.3.65. wctype.h

```

2547
2548 typedef unsigned long wctype_t;
2549 typedef unsigned int wint_t;
2550 typedef const int32_t *wctrans_t;
2551 typedef struct
2552 {
2553     int count;
2554     wint_t value;
2555 }
2556 __mbstate_t;
2557
2558 typedef __mbstate_t mbstate_t;

```

1.3.66. wordexp.h

```

2559
2560 enum
2561 {
2562     WRDE_DOOFFS, WRDE_APPEND, WRDE_NOCMD, WRDE_REUSE, WRDE_SHOWERR, WRDE_UNDEF,
2563     __WRDE_FLAGS
2564 }
2565 ;
2566
2567 typedef struct
2568 {
2569     int we_wordc;
2570     char **we_wordv;
2571     int we_offs;
2572 }
2573 wordexp_t;
2574
2575 enum
2576 {
2577     WRDE_NOSYS, WRDE_NOSPACE, WRDE_BADCHAR, WRDE_BADVAL, WRDE_CMDSUB,

```

```

2578         WRDE_SYNTAX
2579     }
2580     ;

```

1.4. Interface Definitions for libc

2581 The following interfaces are included in libc and are defined by this specification. Unless otherwise noted, these
 2582 interfaces shall be included in the source standard.

2583 Other interfaces listed above for libc shall behave as described in the referenced base document.

_IO_feof

Name

2584 `_IO_feof` — alias for `feof`

Synopsis

```
2585 int _IO_feof(_IO_FILE *__fp);
```

Description

2586 `_IO_feof` tests the end-of-file indicator for the stream pointed to by `__fp`, returning a non-zero value if it is set.

2587 `_IO_feof` is not in the source standard; it is only in the binary standard.

_IO_getc

Name

2588 `_IO_getc` — alias for `getc`

Synopsis

```
2589 int _IO_getc(_IO_FILE *__fp);
```

Description

2590 `_IO_getc` reads the next character from `__fp` and returns it as an unsigned char cast to an int, or EOF on end-of-file
 2591 or error.

2592 `_IO_getc` is not in the source standard; it is only in the binary standard.

_IO_putc

Name

2593 `_IO_putc` — alias for `putc`

Synopsis

2594 `int _IO_putc(int __c, _IO_FILE *__fp);`

Description

2595 `_IO_putc` writes the character `__c`, cast to an unsigned char, to `__fp`.

2596 `_IO_putc` is not in the source standard; it is only in the binary standard.

_IO_puts

Name

2597 `_IO_puts` — alias for `puts`

Synopsis

2598 `int _IO_puts(const char *__c);`

Description

2599 `_IO_puts` writes the string `__s` and a trailing newline to `stdout`.

2600 `_IO_puts` is not in the source standard; it is only in the binary standard.

__assert_fail

Name

2601 `__assert_fail` — abort the program after false assertion

Synopsis

```
2602 void __assert_fail(const char *assertion, const char *file, unsigned int line, const char
2603 *function);
```

Description

2604 The `__assert_fail` receives a string containing function is used to implement the expression `assertion`, assert
 2605 interface of ISO POSIX (2003). The `__assert_fail` function shall print the given `file` filename `file`, and the,
 2606 `line` line number `line`, and prints, `function` function name and a message on the standard error stream in an
 2607 unspecified format, and abort program execution via the abort function. For example:

```
2608     a.c:10: foobar: Assertion a == b failed.
```

2609 If `function` is NULL, `__assert_fail` then aborts program execution via a call to abort. The exact form of the
 2610 message is up to the implementation.

2611 If `function` is NULL, then shall omit information about the function.

2612 `assertion`, `file`, and `line` shall be non-NULL.

2613 The `__assert_fail` function is not in the source standard; it is only in the binary standard. The `assert` interface is
 2614 not in the binary standard; it is only in the source standard. The `assert` may be implemented as a macro.

__ctype_b_loc

Name

2615 `__ctype_b_loc` — accessor function for `__ctype_b` array for ctype functions

Synopsis

```
2616 #include <ctype.h>
2617
2618 extern const unsigned short int **ctype_b_loc (void);
```

Description

2619 `__ctype_b_loc()` returns the address of the array to be used by the ctype functions. This array is locale aware, and
 2620 is local to the current thread if the application is multithreaded.

2621 The `__ctype_b_loc` function shall return a pointer into an array of characters in the current locale that contains
 2622 characteristics for each character in the current character set. The array shall contain a total of 384 characters, and can
 2623 be indexed with any signed or unsigned char (i.e. with an index value between -128 and 255). If the application is
 2624 multithreaded, the array shall be local to the current thread.

2625 This interface is not in the source standard; it is only in the binary standard.

Return Value

2626 The `__ctype_b_loc` function shall return a pointer to the array of characters to be used for the `ctype` family of
 2627 functions (see `<ctype.h>`).

`__ctype_get_mb_cur_max`

Name

2628 `__ctype_get_mb_cur_max` — maximum length of a multibyte character in the current locale

Synopsis

2629 `size_t __ctype_get_mb_cur_max(void);`

Description

2630 `__ctype_get_mb_cur_max` returns the maximum length of a multibyte character in the current locale.

2631 `__ctype_get_mb_cur_max` is not in the source standard; it is only in the binary standard.

`__ctype_tolower_loc`

Name

2632 `__ctype_tolower_loc` — accessor function for `__ctype_b_tolower` array for `ctype tolower()` function

Synopsis

2633 `#include <ctype.h>`
 2634 `int32_t **__ctype_tolower_loc(void);`

Description

2635 The `__ctype_tolower_loc()` ~~returns the address of the array to be used by the `tolower` function.~~ This shall return
 2636 a pointer into an array of characters in the current locale ~~aware~~ that contains lower case equivalents for each character in
 2637 the current character set. The array shall contain a total of 384 characters, ~~and is local to the current thread if~~ can be
 2638 indexed with any signed or unsigned char (i.e. with an index value between -128 and 255). If the application is
 2639 multithreaded, the array shall be local to the current thread.

2640 This interface is not in the source standard; it is only in the binary standard.

__ctype_toupper_loc

Name

2641 `__ctype_toupper_loc` — accessor function for `__ctype_b_toupper` array for `ctype_toupper()` function

Synopsis

```
2642 #include <ctype.h>
2643 int32_t **__ctype_toupper_loc(void);
```

Description

2644 The `__ctype_toupper_loc()` ~~returns the address of the array to be used by the `toupper` function. This shall return~~
 2645 a pointer into an array ~~is~~ characters in the current ~~locale aware~~ that contains upper case equivalents for each character in
 2646 the current character set. The array shall contain a total of 384 characters, ~~and is local to the current thread if~~ can be
 2647 indexed with any signed or unsigned char (i.e. with an index value between -128 and 255). If the application is
 2648 multithreaded, the array shall be local to the current thread.

2649 This interface is not in the source standard; it is only in the binary standard.

__cxa_atexit

Name

2650 `__cxa_atexit` — register a function to be called by `exit` or when a shared library is unloaded

Synopsis

```
2651 | int __cxa_atexit(void (*func) (void *), void *arg, void *dso_handle);
```

Description

2652 `__cxa_atexit` registers a function to be called by `exit` or when a shared library is unloaded. ~~This~~

2653 The `__cxa_atexit` function is ~~only called from code generated by the C++ compiler.~~

2654 ~~`__cxa_atexit` has the same specification as used to implement `atexit`, as described in ISO POSIX (2003). Calling~~

2655 `atexit(func)`

2656 from the statically linked part of an application shall be equivalent to

```
2657 | __cxa_atexit(func, NULL, NULL)
```

2658 .

2659 `__cxa_atexit` is not in the source standard; it is only in the binary standard. `atexit` is not in the binary standard; it
 2660 is only in the source standard.

__daylight

Name

2661 ~~— global variable containing daylight~~
 2662 `__daylight` — Daylight savings time flag

Synopsis

2663 `int __daylight;`

Description

2664 The integer variable `__daylight` shall implement the daylight savings time flag `daylight` as specified in the
 2665 ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~
 2666 header file `<time.h>`.

2667 `__daylight` is not in the source standard; it is only in the binary standard. `daylight` is not in the binary standard; it
 2668 is only in the source standard.

__environ

Name

2669 `__environ` — alias for `environ` - user environment

Synopsis

2670 `extern char **__environ;`

Description

2671 `__environ` is an alias for `environ` - user environment.

2672 `__environ` has the same specification as `environ`.

2673 `__environ` is not in the source standard; it is only in the binary standard.

__errno_location

Name

2674 `__errno_location` — address of `errno` variable

Synopsis

2675 `int *__errno_location(void);`

Description

2676 `__errno_location` is not in the source standard; it is only in the binary standard.

__fpending

Name

2677 `__fpending` — returns in bytes the amount of output pending on a stream

Synopsis

2678 `#include <stdio_ext.h>`
 2679 `size_t __fpending(FILE *stream);`

Description

2680 `__fpending` returns the amount of output in bytes pending on a stream.

2681 `__fpending` is not in the source standard; it is only in the binary standard.

__getpagesize

Name

2682 `__getpagesize` — alias for `getpagesize` - get current page size

Synopsis

2683 `extern int __getpagesize(void);`

Description

2684 `__getpagesize` is an alias for `getpagesize` - get current page size.

2685 `__getpagesize` has the same specification as `getpagesize`.

2686 `__getpagesize` is not in the source standard; it is only in the binary standard.

__getpgid

Name

2687 `__getpgid` — get the process group id

Synopsis

2688 `pid_t __getpgid(pid_t pid);`

Description

2689 `__getpgid` has the same specification as `getpgid`.

2690 `__getpgid` is not in the source standard; it is only in the binary standard.

__h_errno_location

Name

2691 `__h_errno_location` — address of `h_errno` variable

Synopsis

2692 `int *__h_errno_location(void);`

Description

2693 `__h_errno_location` returns the address of the `h_errno` variable, where `h_errno` is as specified in the *Single*
 2694 *Unix Specification*.

2695 `__h_errno_location` is not in the source standard; it is only in the binary standard. Note that `h_errno` itself is only
 2696 in the source standard; it is not in the binary standard.

__isinf

Name

2697 `__isinf` — test for infinity

Synopsis

2698 `int __isinf(double arg);`

Description

2699 `__isinf` has the same specification as `isinf` in the *Single UNIX Specification, Version 3*, except that the argument
2700 type for `__isinf` is known to be double.

2701 `__isinf` is not in the source standard; it is only in the binary standard.

__isinf

Name

2702 `__isinf` — test for infinity

Synopsis

2703 `int __isinf(float arg);`

Description

2704 `__isinf` has the same specification as `isinf` in the *Single UNIX Specification, Version 3*, except that the argument
2705 type for `__isinf` is known to be float.

2706 `__isinf` is not in the source standard; it is only in the binary standard.

__isinfl

Name

2707 `__isinfl` — test for infinity

Synopsis

2708 `int __isinfl(long double arg);`

Description

2709 `__isinfl` has the same specification as `isinf` in the *Single UNIX Specification, Version 3*, except that the argument
 2710 type for `__isinfl` is known to be long double.

2711 `__isinfl` is not in the source standard; it is only in the binary standard.

__isnan

Name

2712 `__isnan` — test for infinity

Synopsis

2713 `int __isnan(double arg);`

Description

2714 `__isnan` has the same specification as `isnan` in the *Single UNIX Specification, Version 3*, except that the argument
 2715 type for `__isnan` is known to be double.

2716 `__isnan` is not in the source standard; it is only in the binary standard.

__isnanf

Name

2717 `__isnanf` — test for infinity

Synopsis

2718 `int __isnanf(float arg);`

Description

2719 `__isnanf` has the same specification as `isnan` in the *Single UNIX Specification, Version 3*, except that the argument
 2720 type for `__isnanf` is known to be float.

2721 `__isnanf` is not in the source standard; it is only in the binary standard.

__isnansl

Name

2722 `__isnansl` — test for infinity

Synopsis

2723 `int __isnansl(long double arg);`

Description

2724 `__isnansl` has the same specification as `isnan` in the *Single UNIX Specification, Version 3*, except that the argument
 2725 type for `__isnansl` is known to be long double.

2726 `__isnansl` is not in the source standard; it is only in the binary standard.

__libc_current_sigrtmax

Name

2727 `__libc_current_sigrtmax` — return number of available real-time signal with lowest priority

Synopsis

2728 `int __libc_current_sigrtmax(void);`

Description

2729 `__libc_current_sigrtmax` returns the number of an available real-time signal with the lowest priority.

2730 `__libc_current_sigrtmax` is not in the source standard; it is only in the binary standard.

__libc_current_sigrtmin

Name

2731 `__libc_current_sigrtmin` — return number of available real-time signal with highest priority

Synopsis

2732 `int __libc_current_sigrtmin(void);`

Description

2733 `__libc_current_sigrtmin` returns the number of an available real-time signal with the highest priority.

2734 `__libc_current_sigrtmin` is not in the source standard; it is only in the binary standard.

__libc_start_main

Name

2735 `__libc_start_main` — initialization routine

Synopsis

2736 `__libc_start_main`(int (*main) (int, char**, char**), int argc, char *__unbounded
2737 *__unbounded ubp_av, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void),
2738 void (*__unbounded stack_end));

Description

2739 The `__libc_start_main` ~~initializes glibc~~ function shall initialize the process, call the *main* function with
2740 appropriate arguments, and handle the return from *main*.

2741 `__libc_start_main` is not in the source standard; it is only in the binary standard.

__lxstat

Name

2742 `__lxstat` — inline wrapper around call to `lxstat`

Synopsis

2743 `#include <ctype.h>`
2744 `int __lxstat`(int version, char *__path, (struct stat *__statbuf));

Description

2745 `__lxstat` is an inline wrapper around call to `lxstat`.

2746 `__lxstat` is not in the source standard; it is only in the binary standard.

__mempcpy

Name

2747 `__mempcpy` — copy given number of bytes of source to destination

Synopsis

```
2748 #include <string.h>
2749 | extern ptr_t __mempcpy(ptr_t restrict dest, const ptr_t restrict src, size_t n);
```

Description

2750 `__mempcpy` copies *n* bytes of source to destination, returning pointer to bytes after the last written byte.

2751 `__mempcpy` is not in the source standard; it is only in the binary standard.

__rawmemchr

Name

2752 `__rawmemchr` — scan memory

Synopsis

```
2753 #include <string.h>
2754 | extern ptr_t __rawmemchr(const ptr_t s, int c);
```

Description

2755 `__rawmemchr` searches in *s* for *c*.

2756 `__rawmemchr` is a weak alias to `rawmemchr`. It is similar to `memchr`, but it has no length limit.

2757 `__rawmemchr` is not in the source standard; it is only in the binary standard.

__register_atfork

Name

2758 `__register_atfork` — alias for `register_atfork`

Synopsis

2759 `int __register_atfork(void (*prepare)(), void (*parent)(), void (*child)(), void`
 2760 `*__dso_handle);`

Description

2761 `__register_atfork` implements `pthread_atfork` as specified in ISO/IEC 9945: POSIX (2003-Portable
 2762 Operating System (POSIX) and The Single UNIX® Specification (SUS) V3). The additional parameter
 2763 `__dso_handle` allows a shared object to pass in its handle so that functions registered by `__register_atfork`
 2764 can be unregistered by the runtime when the shared object is unloaded.

__sigsetjmp

Name

2765 `__sigsetjmp` — save stack context for non-local goto

Synopsis

2766 `int __sigsetjmp(jmp_buf env, int savemask);`

Description

2767 `__sigsetjmp` has the same behavior as `sigsetjmp` as specified by the ~~Single UNIX Specification, Version 2~~ ISO
 2768 POSIX (2003).
 2769 `__sigsetjmp` is not in the source standard; it is only in the binary standard.

__strcpy

Name

2770 `__strcpy` — copy a string returning a pointer to its end

Synopsis

2771 `#include <string.h>`
 2772 `char *__strcpy(char *dest, const char *src);`

Description

2773 `__strcpy` copies the string *src* (including the terminating `/0` character) to the array *dest*. The strings may not
 2774 overlap, and *dest* must be large enough to receive the copy.

Return Value

2775 `__strcpy` returns a pointer to the end of the string *dest* (that is, the address of the terminating `NULL` character) rather
 2776 than the beginning.
 2777 `__strcpy` has the same specification as `strcpy`.
 2778 `__strcpy` is not in the source standard; it is only in the binary standard.

__strdup

Name

2779 `__strdup` — alias for `strdup`

Synopsis

2780 `char *__strdup(const char string);`

Description

2781 `__strdup` has the same specification as `strdup`.
 2782 `__strdup` is not in the source standard; it is only in the binary standard.

__strtod_internal

Name

2783 `__strtod_internal` — underlying function for `strtod`

Synopsis

2784 `double __strtod_internal(const char *__nptr, char **__endptr, int __group);`

Description

2785 `__group` shall be 0 or the behavior of `__strtod_internal` is undefined.

2786 `__strtod_internal(__nptr, __endptr, 0)` has the same specification as `strtod(__nptr, __endptr)`.

2787 `__strtod_internal` is not in the source standard; it is only in the binary standard.

__strtof_internal

Name

2788 `__strtof_internal` — underlying function for `strtof`

Synopsis

2789 `float __strtof_internal(const char *__nptr, char **__endptr, int __group);`

Description

2790 `__group` shall be 0 or the behavior of `__strtof_internal` is undefined.

2791 `__strtof_internal(__nptr, __endptr, 0)` has the same specification as `strtof(__nptr, __endptr)`.

2792 `__strtof_internal` is not in the source standard; it is only in the binary standard.

__strtok_r

Name

2793 `__strtok_r` — alias for `strtok_r`

Synopsis

2794 `char *__strtok_r(char *__restrict s, __const char *__restrict delim, char **__restrict`
 2795 `save_ptr);`

Description

2796 `__strtok_r` has the same specification as `strtok_r`.

2797 `__strtok_r` is not in the source standard; it is only in the binary standard.

__strtol_internal

Name

2798 `__strtol_internal` — alias for `strtol`

Synopsis

2799 `long int __strtol_internal(const char *__nptr, char **__endptr, int __base, int __group);`

Description

2800 `__group` shall be 0 or the behavior of `__strtol_internal` is undefined.

2801 `__strtol_internal(__nptr, __endptr, __base, 0)` has the same specification as `strtol(__nptr,`
 2802 `__endptr, __base)`.

2803 `__strtol_internal` is not in the source standard; it is only in the binary standard.

__strtold_internal

Name

2804 `__strtold_internal` — underlying function for `strtold`

Synopsis

2805 `long double __strtold_internal(const char *__nptr, char **__endptr, int __group);`

Description

2806 `__group` shall be 0 or the behavior of `__strtold_internal` is undefined.

2807 `__strtold_internal(__nptr, __endptr, 0)` has the same specification as `strtold(__nptr, __endptr)`.

2808 `__strtold_internal` is not in the source standard; it is only in the binary standard.

__strtoll_internal

Name

2809 `__strtoll_internal` — underlying function for `strtoll`

Synopsis

2810 `long long __strtoll_internal(const char *__nptr, char **__endptr, int __base, int __group);`

Description

2811 `__group` shall be 0 or the behavior of `__strtoll_internal` is undefined.

2812 `__strtoll_internal(__nptr, __endptr, __base, 0)` has the same specification as `strtoll(__nptr, __endptr, __base)`.

2814 `__strtoll_internal` is not in the source standard; it is only in the binary standard.

__strtoul_internal

Name

2815 `__strtoul_internal` — underlying function for `strtoul`

Synopsis

2816 `unsigned long int __strtoul_internal(const char *__nptr, char **__endptr, int __base, int`
 2817 `__group);`

Description

2818 `__group` shall be 0 or the behavior of `__strtoul_internal` is undefined.

2819 `__strtoul_internal(__nptr, __endptr, __base, 0)` has the same specification as `strtoul(__nptr,`
 2820 `__endptr, __base)`.

2821 `__strtoul_internal` is not in the source standard; it is only in the binary standard.

__strtoull_internal

Name

2822 `__strtoull_internal` — underlying function for `strtoull`

Synopsis

2823 `unsigned long long __strtoull_internal(const char *__nptr, char **__endptr, int __base,`
 2824 `int __group);`

Description

2825 `__group` shall be 0 or the behavior of `__strtoull_internal` is undefined.

2826 `__strtoull_internal(__nptr, __endptr, __base, 0)` has the same specification as `strtoull(__nptr,`
 2827 `__endptr, __base)`.

2828 `__strtoull_internal` is not in the source standard; it is only in the binary standard.

__sysconf

Name

2829 `__sysconf` — get configuration information at runtime

Synopsis

2830 `#include <unistd.h>`
 2831 `long __sysconf(int name);`

Description

2832 `__sysconf` gets configuration information at runtime.
 2833 `__sysconf` is weak alias to `sysconf`.
 2834 `__sysconf` has the same specification as `sysconf`.
 2835 `__sysconf` is not in the source standard; it is only in the binary standard.

__sysv_signal

Name

2836 `__sysv_signal` — signal handling

Synopsis

2837 `__sighandler_t __sysv_signal(int sig, __sighandler_t handler);`

Description

2838 `__sysv_signal` has the same behavior as `signal` as specified by ~~X/Open~~ISO POSIX (2003).
 2839 `__sysv_signal` is not in the source standard; it is only in the binary standard.

__timezone

Name

2840 — global variable containing timezone

Synopsis

2841 `long int __timezone;`

Description

2842 | `__timezone` has the same specification as `timezone` in the *Single UNIX Specification, Version 3* ISO POSIX (2003).

__tzname

Name

2843 — global variable containing the timezone

Synopsis

2844 `char *__tzname[2];`

Description

2845 | `__tzname` has the same specification as `tzname` in the *Single UNIX Specification, Version 3* ISO POSIX (2003).

2846 | Note that the array size of 2 is explicit in the *Single UNIX Specification, Version 3* ISO POSIX (2003), but not in the
2847 | *Single UNIX Specification, Version 2* SUSv2.

__wcstod_internal

Name

2848 `__wcstod_internal` — underlying function for `wcstod`

Synopsis

2849 `double __wcstod_internal(const wchar_t *nptr, wchar_t **endptr, int group);`

Description

2850 *group* shall be 0 or the behavior of `__wcstod_internal` is undefined.

2851 `__wcstod_internal(nptr, endptr, 0)` has the same specification as `wcstod(nptr, endptr)`.

2852 `__wcstod_internal` is not in the source standard; it is only in the binary standard.

__wstof_internal

Name

2853 `__wstof_internal` — underlying function for `wstof`

Synopsis

2854 `float __wstof_internal(const wchar_t *nptr, wchar_t **endptr, int group);`

Description

2855 *group* shall be 0 or the behavior of `__wstof_internal` is undefined.

2856 `__wstof_internal(nptr, endptr, 0)` has the same specification as `wstof(nptr, endptr)`.

2857 `__wstof_internal` is not in the source standard; it is only in the binary standard.

__wcstol_internal

Name

2858 `__wcstol_internal` — underlying function for `wcstol`

Synopsis

2859 `long __wcstol_internal(const wchar_t *nptr, wchar_t **endptr, int base, int group);`

Description

2860 *group* shall be 0 or the behavior of `__wcstol_internal` is undefined.

2861 `__wcstol_internal(nptr, endptr, base, 0)` has the same specification as `wcstol(nptr, endptr, base)`.

2862 `__wcstol_internal` is not in the source standard; it is only in the binary standard.

__wcstold_internal

Name

2863 `__wcstold_internal` — underlying function for `wcstold`

Synopsis

2864 `long double __wcstold_internal(const wchar_t *nptr, wchar_t **endptr, int group);`

Description

2865 *group* shall be 0 or the behavior of `__wcstold_internal` is undefined.

2866 `__wcstold_internal(nptr, endptr, 0)` has the same specification as `wcstold(nptr, endptr)`.

2867 `__wcstold_internal` is not in the source standard; it is only in the binary standard.

__wcstoul_internal

Name

2868 `__wcstoul_internal` — underlying function for `wcstoul`

Synopsis

2869 `unsigned long __wcstoul_internal(const wchar_t *restrict nptr, wchar_t **restrict endptr,`
2870 `int base, int group);`

Description

2871 *group* shall be 0 or the behavior of `__wcstoul_internal` is undefined.

2872 `__wcstoul_internal(nptr, endptr, base, 0)` has the same specification as `wcstoul(nptr, endptr,`
2873 `base)`.

2874 `__wcstoul_internal` is not in the source standard; it is only in the binary standard.

__xmknod

Name

2875 `__xmknod` — make block or character special file

Synopsis

2876 `int __xmknod(int ver, __const char *path, __mode_t mode, __dev_t *dev);`

Description

2877 ~~ver shall be 1 or the behavior of __xmknod is undefined.~~

2878 The `__xmknod` shall implement the `mknod` interface from ISO POSIX (2003).

2879 `__xmknod(1, path, mode, dev)` has the same specification as `mknod(path, mode, dev)`.

2880 ~~Note that ver shall be 1 or the format behavior of dev_t is not the same as the argument that the kernel syscall uses.~~

2881 `__xmknod` is undefined.

2882 The `__xmknod` function is not in the source standard; it is only in the binary standard. The `mknod` function is not in the

2883 binary standard; it is only in the source standard.

__xstat

Name

2884 `__xstat` — ~~provide inode information~~Get File Status

Synopsis

2885 `#include <sys/stat.h>`

```

2886 #include <unistd.h>
2887 int __xstat(int __ver, const char *__filenamepath, (struct stat *__stat_buf));
2888 int __lxstat(int __ver, const char *__filenamepath, (struct stat *__stat_buf));
2889 int __fxstat(int __ver, int __filedesc, fildes, (struct stat *__stat_buf));

```

Description

```

2890 __ver shall be 3 or the behavior of these functions is undefined.
2891 __filename is as specified in POSIX.
2892 __filedesc is as specified in POSIX.
2893 __stat_buf is as specified in POSIX.
2894 __xstat(3, __filename, __stat_buf) has the same specification as stat(__filename, __stat_buf) as
2895 specified by POSIX.
2896 __lxstat(3, __filename, __stat_buf) has the same specification as lstat(__filename, __stat_buf) as
2897 specified by POSIX.
2898 __fxstat(3, __filedesc, __stat_buf) has the same specification as fstat(__filedesc, __stat_buf) as
2899 specified by POSIX.
2900 Note that the struct stat used by these functions is not the one that the kernel uses.
2901 The functions __xstat, __lxstat, and __fxstat shall implement the ISO POSIX (2003) functions stat, lstat,
2902 and fstat respectively.
2903 ver shall be 3 or the behavior of these functions is undefined.
2904 __xstat(3, path, stat_buf) shall behave as stat(path, stat_buf) as specified by ISO POSIX (2003).
2905 __lxstat(3, path, stat_buf) shall behave as lstat(path, stat_buf) as specified by ISO POSIX (2003).
2906 __fxstat(3, fildes, stat_buf) shall behave as fstat(fildes, stat_buf) as specified by ISO POSIX
2907 (2003).
2908 __xstat, __lxstat, and __fxstat are not in the source standard; they are only in the binary standard.
2909 stat, lstat, and fstat are not in the binary standard; they are only in the source standard.

```

__xstat64

Name

```

2910 __xstat64 — provide inode information Get File Status

```

Synopsis

```

2911 #define _LARGEFILE_SOURCE 1
2912 #include <sys/stat.h>

```

```

2913 #include <unistd.h>
2914 int __xstat64(int __ver, const char *__filenamepath, (struct stat64 *__stat_buf));
2915 int __lxstat64(int __ver, const char *__filenamepath, (struct stat64 *__stat_buf));
2916 int __fxstat64(int __ver, int __filedesc, (struct stat64 *__stat_buf));

```

Description

```

2917 __ver shall be 3 or the behavior of these functions is undefined.
2918 __filename is as specified by the Large File Summit.
2919 __filedesc is as specified by the Large File Summit.
2920 __stat_buf is as specified by the Large File Summit.
2921 __xstat64(3, __filename, __stat_buf) has the same specification as stat64(__filename, __stat_buf)
2922 as specified by the Large File Summit.
2923 __lxstat64(3, __filename, __stat_buf) has the same specification as lstat64(__filename,
2924 __stat_buf) as specified by the Large File Summit.
2925 __fxstat64(3, __filedesc, __stat_buf) has the same specification as fstat64(__filedesc,
2926 __stat_buf) as specified by the Large File Summit.
2927 The functions __xstat64, __lxstat64, and __fxstat64 shall implement the Large File Support functions
2928 stat64, lstat64, and fstat64 respectively.
2929 ver shall be 3 or the behavior of these functions is undefined.
2930 __xstat64(3, path, stat_buf) shall behave as stat(path, stat_buf) as specified by Large File Support.
2931 __lxstat64(3, path, stat_buf) shall behave as lstat(path, stat_buf) as specified by Large File Support.
2932 __fxstat64(3, fildes, stat_buf) shall behave as fstat(fildes, stat_buf) as specified by Large File
2933 Support.
2934 __xstat64, __lxstat64, and __fxstat64 are not in the source standard; they are only in the binary standard.
2935 stat64, lstat64, and fstat64 are not in the binary standard; they are only in the source standard.

```

__environ

Name

```

2936 __environ — alias for environ - user environment

```

Synopsis

```

2937 extern char **_environ;

```

Description

```

2938 __environ is an alias for environ - user environment.

```

_nl_msg_cat_cntr

Name

2939 `_nl_msg_cat_cntr` — new catalog load counter

Synopsis

```
2940 #include <libintl.h>
2941
2942 extern int _nl_msg_cat_cntr;
```

Description

2943 `_nl_msg_cat_cntr` is incremented each time a new catalog is loaded. It is a variable defined in `loadmsgcat.c`
 2944 and is used by Message catalogs for internationalization.

_obstack_begin

Name

2945 `_obstack_begin` — initialize an obstack for use

Synopsis

```
2946 #include <obstack.h>
2947 | extern int _obstack_begin(struct obstack *, int, int, void *(*)(long), void (*)(void *));
```

Description

2948 `_obstack_begin` initializes an obstack for use.

Future Directions

2949 | Future versions of this specification may not include support for this interface.

`_obstack_newchunk`

Name

2950 `_obstack_newchunk` — allocate a new current chunk of memory for the obstack

Synopsis

```
2951 #include <obstack.h>
2952 | extern void _obstack_newchunk(struct obstack *, int);
```

Description

2953 `_obstack_newchunk` allocates a new current chunk of memory for the obstack.

Future Directions

2954 | Future versions of this specification may not include support for this interface.

`_sys_errlist`

Name

2955 `_sys_errlist` — array containing the "C" locale strings used by `strerror()`

Synopsis

```
2956 #include <stdio.h>
2957
2958 extern const char *const _sys_errlist[];
```

Description

2959 `_sys_errlist` is an array containing the "C" locale strings used by `strerror`. This normally should not be used
2960 directly. `strerror` provides all of the needed functionality.

`_sys_siglist`

Name

2961 `_sys_siglist` — array containing the names of the signal names

Synopsis

```
2962 #include <signal.h>
2963
```

```
2964 extern const char *const _sys_siglist[NSIG];
```

Description

```
2965 _sys_siglist is an array containing the names of the signal names.
```

```
2966 | The _sys_siglist existsarray is only for compatibility; in the binary standard; it is not in the source standard.
```

```
2967 | Applications wishing to access the names of signals should use the strsignal instead. (See string.h)function.
```


acct

Name

2968 `acct` — switch process accounting on or off

Synopsis

```
2969 #include <dirent.h>
2970 int acct(const char *filename);
```

Description

2971 When *filename* is the name of an existing file, `acct` turns accounting on and appends a record to *filename* for
 2972 each terminating process. When *filename* is `NULL`, `acct` turns accounting off.

Return Value

2973 On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

2974 `ENOSYS`

2975 BSD process accounting has not been enabled when the operating system kernel was compiled. The kernel
 2976 configuration parameter controlling this feature is `CONFIG_BSD_PROCESS_ACCT`.

2977 `ENOMEM`

2978 Out of memory.

2979 `EPERM`

2980 The calling process has no permission to enable process accounting.

2981 `EACCES`

2982 *filename* is not a regular file.

2983 `EIO`

2984 Error writing to the *filename*.

2985 `EUSERS`

2986 There are no more free file structures or we run out of memory.

adjtime

Name

2987 `adjtime` — correct the time to allow synchronization of the system clock

Synopsis

```
2988 #include <time.h>
2989 int adjtime((const struct timeval *delta), (struct timeval *olddelta));
```

Description

2990 `adjtime` makes small adjustments to the system time as returned by `gettimeofday(2)`, advancing or retarding it by
 2991 the time specified by the `timeval` *delta*. If *delta* is negative, the clock is slowed down by incrementing it more
 2992 slowly than normal until the correction is complete. If *delta* is positive, a larger increment than normal is used. The
 2993 skew used to perform the correction is generally a fraction of one percent. Thus, the time is always a monotonically
 2994 increasing function. A time correction from an earlier call to `adjtime` may not be finished when `adjtime` is called
 2995 again. If *olddelta* is non-NULL, the structure pointed to will contain, upon return, the number of microseconds still
 2996 to be corrected from the earlier call.

2997 `adjtime` may be used by time servers that synchronize the clocks of computers in a local area network. Such time
 2998 servers would slow down the clocks of some machines and speed up the clocks of others to bring them to the average
 2999 network time.

3000 The `adjtime` is restricted to the super-user.

Return Value

3001 On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

3002 EFAULT

3003 An argument points outside the process's allocated address space.

3004 EPERM

3005 The process's effective user ID is not that of the super-user.

adjtimex

Name

3006 adjtimex — tune kernel clock (DEPRECATED)

Synopsis

```
3007 #include <sys/timex.h>
3008 int adjtimex((struct timex *buf));
```

Description

3009 The adjtimex function is deprecated from the LSB and is expected to disappear from a future version of the LSB.⁺

3010 The LSB generally does not include interfaces unlikely to be used by software applications.

3011 Linux uses David L. Mills' clock adjustment algorithm (see *RFC 1305*). adjtimex reads and optionally sets
3012 adjustment parameters for this algorithm. adjtimex takes a pointer to a timex structure, updates kernel parameters
3013 from field values, and returns the same structure with current kernel values. This structure is declared as follows:

```
3014 struct timex {
3015     int modes;           /* mode selector */
3016     long offset;         /* time offset (usec) */
3017     long freq;           /* frequency offset (scaled ppm) */
3018     long maxerror;       /* maximum error (usec) */
3019     long esterror;       /* estimated error (usec) */
3020     int status;          /* clock command/status */
3021     long constant;       /* pll time constant */
3022     long precision;      /* clock precision (usec) (read only) */
3023     long tolerance;      /* clock frequency tolerance (ppm)
3024                          (read only) */
3025     struct timeval time; /* current time (read only) */
3026     long tick;           /* usecs between clock ticks */
3027 };
```

3028 *modes* determines which parameters, if any, to set. *modes* may contain a bitwise-or combination of zero or more of
3029 the following bits:

```
3030 #define ADJ_OFFSET      0x0001 /* time offset */
3031 #define ADJ_FREQUENCY   0x0002 /* frequency offset */
3032 #define ADJ_MAXERROR    0x0004 /* maximum time error */
3033 #define ADJ_ESTERROR    0x0008 /* estimated time error */
3034 #define ADJ_STATUS      0x0010 /* clock status */
3035 #define ADJ_TIMECONST   0x0020 /* pll time constant */
3036 #define ADJ_TICK        0x4000 /* tick value */
```

```
3037     #define ADJ_OFFSET_SINGLESOT 0x8001 /* old-fashioned adjtime */
```

3038 Ordinary users are restricted to a 0 value for *modes*. Only the superuser may set any parameters.

Return Value

3039 On success, *adjtimex* returns the clock state:

```
3040     #define TIME_OK      0 /* clock synchronized */
3041     #define TIME_INS     1 /* insert leap second */
3042     #define TIME_DEL     2 /* delete leap second */
3043     #define TIME_OOP     3 /* leap second in progress */
3044     #define TIME_WAIT    4 /* leap second has occurred */
3045     #define TIME_BAD     5 /* clock not synchronized */
```

3046 On error, the global variable *errno* is set to -1.

Errors

3047 EFAULT

3048 *buf* does not point to writable memory.

3049 EPERM

3050 *buf.mode* is nonzero and the user is not super-user.

3051 EINVAL

3052 An attempt is made to set *buf.offset* to a value outside of the range -131071 to +131071, or to set
 3053 *buf.status* to a value other than those listed above, or to set *buf.tick* to a value outside of the range
 3054 900000/HZ to 1100000/HZ, where HZ is the system timer interrupt frequency.

Notes

3056 1. The LSB generally does not include interfaces unlikely to be used by software applications.

asprintf

Name

3057 | `asprintf` — write formatted output to a string dynamically allocated with `malloc` and store the address of the string

Synopsis

```
3058 | #include <stdio.h>
3059 | extern int asprintf(char ** restrict ptr, const char * restrict format ...);
```

Description

3060 | ~~`asprintf` has the same behavior as `sprintf`, but calls `malloc` to dynamically allocate space for the output, and then~~
 3061 | ~~puts the output string in that space.~~

3062 | ~~`asprintf` stores the address of the string in `ptr`.~~

3063 | The `asprintf` function shall behave as `sprintf`, except that the output string shall be dynamically allocated space
 3064 | of sufficient length to hold the resulting string. The address of this dynamically allocated string shall be stored in the
 3065 | location referenced by `ptr`.

Return Value

3066 | Refer to `fprintf`.

Errors

3067 | Refer to `fprintf`.

bind_textdomain_codeset

Name

3068 | `bind_textdomain_codeset` — specify encoding for message retrieval from message catalog for domain
 3069 | `DOMAINNAME`

Synopsis

```
3070 | #include <libintl.h>
3071 |
```

```
3072 | extern char * bind_textdomain_codeset (const char * domainname , const char * codeset );
```

Description

3073 The *bind_textdomain_codeset* function can be used to specify the output codeset for message catalogs for domain
 3074 *domainname*. The *codeset* argument shall be a valid codeset name which can be used for the *iconv_open()*
 3075 ~~function~~ function, or a null pointer. If the *codeset* argument is the null pointer, then function returns the currently
 3076 selected codeset for the domain with the name *domainname*. It ~~returns~~ shall return a null pointer if no codeset has yet
 3077 been selected

3078 Each successive call to *bind_textdomain_codeset* function overrides the settings made by the preceding call
 3079 with the same *domainname*.

3080 The *bind_textdomain_codeset* function ~~can~~ shall return a pointer to a string containing the name of the selected
 3081 codeset. The string shall be used several times. ~~If used multiple times, with the same *domainname* argument, the later~~
 3082 ~~call overrides the settings made by the earlier one~~ allocated internally in the function and shall not be changed or freed
 3083 by the user.

3084 The *bind_textdomain_codeset* function returns a pointer to a string containing the name of the selected codeset.
 3085 The string is allocated internally in the function and shall not be changed by the user.

Parameters

3086 *domainname*

3087 The *domainname* argument is applied to the ~~currently~~ currently active LC_MESSAGE locale. It is equivalent
 3088 in syntax and meaning to the *domainname* argument to *textdomain()*, except that the selection of the
 3089 domain is valid only for the duration of the call.

Return

3090 ~~Returns the currently selected codeset name. It returns null pointer if no codeset has yet been selected.~~

Errors

3091 ~~The function is not required to set~~ name of the output codeset for the ~~external~~ selected domain, or NULL to select
 3092 the current codeset.

3093 If *domainname* is the null pointer, or is an empty string, *bind_textdomain_codeset* shall fail, but need not
 3094 set *errno* variable.

Return Value

3095 Returns the currently selected codeset name. It returns a null pointer if no codeset has yet been selected.

Errors

3096 ENOMEM

3097 Insufficient memory available to allocate return value.

See Also

3098 | gettext (baselib-gettext.html), dgettext, ngettext, dngettext, dcgettext, dcngettext, textdomain, bindtextdomain,
 3099 | ~~bind_textdomain_codeset~~

bindresvport

Name

3100 | bindresvport — bind socket to privileged IP port

Synopsis

```
3101 | #include <sys/types.h>
3102 | #include <rpc/rpc.h>
3103 | int bindresvport(int sd, struct sockaddr_in *sin);
```

Description

3104 | If the process has appropriate privilege, the `bindresvport` ~~binds~~ function shall bind a socket to a privileged IP port.
 3105 | ~~This function can be used only by *root*.~~

Return Value

3106 | On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

3107 | EPERM
 3108 | The process did not have appropriate privilege.

3109 | EPFNOSUPPORT
 3110 | Address of *sin* did not match address family of *sd*.

bindtextdomain

Name

3111 | bindtextdomain — specify the locale location of a message catalog

Synopsis

```
3112 | #include <libintl.h>
3113 | extern char *bindtextdomain(const char *domainname, const char *dirname);
```

Description

3114 | The bindtextdomain shall set the the base directory of the hierarchy containing message catalogs for a given
3115 | message domain.

3116 | The bindtextdomain function specifies that the *domainname* message catalog can be found in the *dirname*
3117 | directory hierarchy, rather than in the system default locale data base.

3118 | ~~bindtextdomain applies domainname to the currently active LC_MESSAGE locale. This usage If dirname is~~
3119 | ~~equivalent in syntax and meaning~~not NULL, the base directory for message catalogs belonging to domain
3120 | *domainname* shall be set to the textdomain function's application of *domainname*, except that the selection of
3121 | the domain in bind_textdomain_codeset *dirname*. If *dirname* is valid only for NULL, the default base
3122 | directory for message catalogs shall not be altered.

3123 | The function shall make copies of the each argument strings as needed.

3124 | *dirname* can be an absolute or relative pathname.

3125 | Applications that wish to use `chdir` should always use absolute pathnames to avoid misadventently selecting the
3126 | wrong or non-existent directory.

3127 | If *domainname* is the null pointer, or is an empty string, bindtextdomain shall fail, but need not set `errno`.

3128 | The bindtextdomain function shall return a pointer to a string containing the name of the selected directory. The
3129 | string shall be allocated internally in the function and shall not be changed or freed by the user.

Return Value

3130 | On success, bindtextdomain ~~returns~~ shall return a pointer to a string containing the directory pathname currently
3131 | bound to the domain. On failure, a NULL pointer is returned, and the global variable `errno` may be set to indicate the
3132 | error.

Errors

3133 | ENOMEM

3134 | Insufficient memory was available.

See Also


```

3135 | gettext (baselib-gettext.html), dgettext, ngettext, dngettext, dcgettext, dcngettext, textdomain, bind_textdomain,
3136 | bind_textdomain_codeset

```

cfmakeraw

Name

```

3137 | cfmakeraw — get and set terminal attributes

```

Synopsis

```

3138 | #include <termios.h>
3139 | void cfmakeraw(struct termios *termios_p);

```

Description

```

3140 | The cfmakeraw(p) function shall set the attributes of the termios structure referenced by termios_p as follows:

```

```

3141 |     termios_p->c_iflag &= ~(IGNBRK|BRKINT|PARMRK|ISTRIP
3142 |                           |INLCR|IGNCR|ICRNL|IXON);
3143 |
3144 |     termios_p->c_oflag &= ~OPOST;
3145 |
3146 |     termios_p->c_lflag &= ~(ECHO|ECHONL|ICANON|ISIG|IEXTEN);
3147 |
3148 |     termios_p->c_cflag &= ~(CSIZE|PARENB);
3149 |
3150 |     termios_p->c_cflag |= CS8;

```

```

3151 | termios_p shall point to a termios structure that contains the following members:

```

```

3152 |     tcflag_t c_iflag;      /* input modes */
3153 |     tcflag_t c_oflag;      /* output modes */
3154 |     tcflag_t c_cflag;      /* control modes */
3155 |     tcflag_t c_lflag;      /* local modes */
3156 |     cc_t c_cc[NCCS];      /* control chars */

```

cfsetspeed

Name

3157 `cfsetspeed` — set terminal input and output data rate

Synopsis

```
3158 #include <termios.h>
3159 int cfsetspeed(struct termios *t, speed_t speed);
```

Description

3160 `cfsetspeed` sets the baud rate values in the `termios` structure. The effects of the function on the terminal as described
 3161 below do not become effective, nor are all errors detected, until the `tcsetattr` function is called. Certain values for
 3162 baud rates set in `termios` and passed to `tcsetattr` have special meanings.

Getting and Setting the Baud Rate

3164 Input and output baud rates are found in the `termios` structure. The unsigned integer `speed_t` is typedef'd in the
 3165 include file `termios.h`. The value of the integer corresponds directly to the baud rate being represented; however, the
 3166 following symbolic values are defined.

```
3167 #define B0      0
3168 #define B50     50
3169 #define B75     75
3170 #define B110    110
3171 #define B134    134
3172 #define B150    150
3173 #define B200    200
3174 #define B300    300
3175 #define B600    600
3176 #define B1200   1200
3177 #define B1800   1800
3178 #define B2400   2400
3179 #define B4800   4800
3180 #define B9600   9600
3181 #define B19200  19200
3182 #define B38400  38400
3183 #ifndef _POSIX_SOURCE
3184 #define EXTA     19200
3185 #define EXTB     38400
```

```
3186      #endif    /*_POSIX_SOURCE */
```

3187 `cfsetspeed` sets both the input and output baud rates in the `termios` structure referenced by `t` to *speed*.

Return Value

3188 On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

3189 EINVAL

3190 Invalid *speed* argument

creat

Name

3191 `creat` — open a file

Description

3192 `creat` is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

3193 May return ENODEV in place of ENXIO

3194 Where the ~~Single UNIX Specification~~ ISO POSIX (2003) specifies an ENXIO return, the implementation may return
3195 either ENXIO or ENODEV. Implementations are ~~encouraged~~ encouraged to return ENXIO.[†]

3196 Notes

3197 †— As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a
3198 kernel patch would be accepted if submitted.

daemon

Name

3199 daemon — run in the background

Synopsis

```
3200 #include <unistd.h>
3201 int daemon(int nochdir, int noclose);
```

Description

3202 The `daemon` function shall create a new process, detached from the controlling terminal. If
 3203 successful, the calling process shall exit and the new process shall continue to execute the application in the
 3204 background as system daemons. Unless, if *nochdir* is non-zero evaluates to true, the current directory shall not be
 3205 changed. Otherwise, `daemon` shall change the current working directory to the root (`/`). Unless if *noclose* is
 3206 non-zero, `daemon` will redirect evaluates to true the standard input, standard output, and standard error file descriptors
 3207 shall not be altered. Otherwise, `daemon` shall close the standard input, standard output and standard error file
 3208 descriptors and reopen them attached to `/dev/null`.

Return Value

3209 On error, -1 is returned, and the global variable `errno` is set to any of the errors specified for the library functions
 3210 `fork(2)` and `setsid(2)`.

dcgettext

Name

3211 `dcgettext` — perform domain and category specific lookup in message catalog for the current `LC_MESSAGES`
 3212 `locale`

Synopsis

```
3213 #include <libintl.h>
```

```

3214 #include <locale.h>
3215 | extern char *dcgettext(const char *domainname, const char *msgid, int category);

```

Description

3216 ~~dcgettext is a domain-specified version of gettext.~~

Parameters

3217 ~~domainname~~

3218 — ~~dcgettext applies domainname to the currently active LC_MESSAGE locale. This usage is equivalent in~~
 3219 ~~syntax and meaning to the textdomain function's application of domainname, except that the selection of the~~
 3220 ~~domain in dcgettext is valid only for the duration of the call.~~

3221 ~~msgid~~

3222 — ~~a NULL terminated string to be matched in the catalogue with respect to a specific domain and the current locale.~~

3223 ~~category~~

3224 — ~~category is used for retrieving messages string for other than LC_MESSAGES category. Available value for~~
 3225 ~~category are LC_CTYPE, LC_COLLATE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME.~~

3226 ~~dcgettext(domainname, msgid, LC_MESSAGES) has the same specification as dgettext(domainname,~~
 3227 ~~msgid). Note that LC_ALL shall not be used.~~

Return Value

3228 ~~On success, the translated NULL terminated string is returned. On error, msgid is returned.~~

Errors

3229 ~~dcgettext will not modify the errno global variable.~~

3230 The dcgettext function is a domain specified version of gettext.

3231 The dcgettext function shall lookup the translation in the current locale of the message identified by *msgid* in the
 3232 domain specified by *domainname* and in the locale category specified by *category*. If *domainname* is NULL,
 3233 the current default domain shall be used. The *msgid* argument shall be a NULL-terminated string to be matched in
 3234 the catalogue. *category* shall specify the locale category to be used for retrieving message strings. The category
 3235 parameter shall be one of *LC_CTYPE*, *LC_COLLATE*, *LC_MESSAGES*, *LC_MONETARY*, *LC_NUMERIC*, or
 3236 *LC_TIME*. The default domain shall not be changed by a call to dcgettext.

Return Value

3237 If a translation was found in one of the specified catalogs, it shall be converted to the current locale's codeset and
 3238 returned. The resulting NULL-terminated string shall be allocated by the dcgettext function, and must not be modified
 3239 or freed. If no translation was found, or category was invalid, *msgid* shall be returned.

Errors

3240 dcgettext shall not modify the errno global variable.

See Also

3241 | gettext (baselib-gettext.html), dgettext, ngettext, dngettext, ~~degettext~~, dcngettext, textdomain, bindtextdomain,
 3242 | bind_textdomain_codeset

dcngettext**Name**

3243 | dcngettext — perform domain and category specific lookup in message catalog for the current LC_MESSAGES
 3244 | locale with plural

Synopsis

3245 | #include <libintl.h>

```

3246 #include <locale.h>
3247 | extern char *dcngettext(const char *domainname, const char *msgid1, const char *msgid2,
3248 unsigned long int n, int category);

```

Description

3249 ~~dcngettext is a plural version of dgettext. (See dgettext for more information.)~~

Parameters

3250 domainname

3251 — ~~dcngettext applies domainname to the currently active LC_MESSAGE locale. This usage is equivalent in~~
 3252 ~~syntax and meaning to the textdomain function's application of domainname, except that the selection of the~~
 3253 ~~domain in dcngettext is valid only for the duration of the call.~~

3254 msgid1

3255 — ~~a NULL terminated string to be matched in the catalogue with respect to a specific domain and the current locale.~~
 3256 ~~If the value of n is 1 and no message catalogs containing a translation for msgid1 are found, msgid1 is~~
 3257 ~~returned.~~

3258 msgid2

3259 — ~~a NULL terminated string to be returned if the value of n is not 1 and no message catalogs are found.~~

3260 n

3261 — ~~determines which plural form is returned, in a language and message catalog dependent way.~~

3262 category

3263 — ~~category is used for retrieving messages string for other than LC_MESSAGES category. Available value for~~
 3264 ~~category are LC_CTYPE, LC_COLLATE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME.~~

3265 ~~dcngettext(domainname, msgid1, msgid2, n, LC_MESSAGES) has the same specification as~~
 3266 ~~dgettext(domainname, msgid1, msgid2, n). Note that LC_ALL shall not be used.~~

Return Value

3267 ~~On success of a msgid1 query, the translated NULL terminated string is returned. On error, the original msgid1 or~~
 3268 ~~msgid2 is returned, according to n.~~

Errors

3269 ~~dcngettext will not modify the errno global variable.~~

3270 The `dcngettext` function is a domain specific version of `gettext`, capable of returning either a singular or plural form
 3271 of the message. The `dcngettext` function shall lookup the translation in the current locale of the message identified
 3272 by *msgid1* in the domain specified by *domainname* and in the locale category specified by *category*. If
 3273 *domainname* is NULL, the current default domain shall be used. The *msgid1* argument shall be a
 3274 NULL-terminated string to be matched in the catalogue. *category* shall specify the locale category to be used for
 3275 retrieving message strings. The *category* parameter shall be one of *LC_CTYPE*, *LC_COLLATE*, *LC_MESSAGES*,
 3276 *LC_MONETARY*, *LC_NUMERIC*, or *LC_TIME*. The default domain shall not be changed by a call to `dcgettext`. If *n*
 3277 is 1 then the singular version of the message is returned, otherwise one of the plural forms is returned, depending on
 3278 the value of *n* and the current locale settings.

Return Value

3279 If a translation corresponding to the value of *n* was found in one of the specified catalogs for *msgid1*, it shall be
 3280 converted to the current locale's codeset and returned. The resulting NULL-terminated string shall be allocated by the
 3281 `dcngettext` function, and must not be modified or freed. If no translation was found, or *category* was invalid,
 3282 *msgid1* shall be returned if *n* has the value 1, otherwise *msgid2* shall be returned.

Errors

3283 `dcngettext` shall not modify the `errno` global variable.

See Also

3284 `gettext` (baselib-gettext.html), `dgettext`, `ngettext`, `dngettext`, `dcgettext`, ~~`dgettext`~~, `textdomain`, `bindtextdomain`,
 3285 `bind_textdomain_codeset`

dgettext

Name

3286 dgettext — perform lookup in message catalog for the current LC_MESSAGES locale

Synopsis

```
3287 #include <libintl.h>
3288 | extern char *dgettext(const char *domainname, const char *msgid);
```

Description

3289 dgettext is a domain specified version of gettext.

Parameters

3290 domainname

3291 dgettext applies *domainname* to the currently active LC_MESSAGE locale. This usage is equivalent in
 3292 syntax and meaning to the `textdomain` function's application of *domainname*, except that the selection of the
 3293 domain in dgettext is valid only for the duration of the call.

3294 msgid

3295 a NULL-terminated string to be matched in the catalogue with respect to a specific domain and the current locale.

Return Value

3296 On success of a *msgid* query, the translated NULL-terminated string is returned. On error, the original *msgid* is
 3297 returned. The length of the string returned is undetermined until dgettext is called.

Errors

3298 dgettext will not modify the `errno` global variable.

See Also

3299 gettext (baselib-gettext.html), dgettext, ngettext, dngettext, dcgettext, dcngettext, textdomain, bindtextdomain,
 3300 bind_textdomain_codeset

dngettext

Name

3301 | `dngettext` — perform lookup in message catalog for the current `LC_MESSAGES` locale

Synopsis

```
3302 | #include <libintl.h>
3303 | extern char *dngettext(const char *domainname, const char *msgid1, const char *msgid2,
3304 | unsigned long int n);
```

Description

3305 | ~~`dngettext` is a plural version of `dgettext`. (See `dgettext` for more information.)~~

Parameters

3306 | `domainname`

3307 | — ~~`dngettext` applies `domainname` to the currently active `LC_MESSAGE` locale. This usage is equivalent in~~
 3308 | ~~syntax and meaning to the `textdomain` function's application of `domainname`, except that the selection of the~~
 3309 | ~~domain in `dngettext` is valid only for the duration of the call.~~

3310 | `msgid1`

3311 | — ~~a NULL terminated string to be matched in the catalogue with respect to a specific domain and the current locale.~~
 3312 | ~~If the value of `n` is 1 and no message catalogs containing a translation for `msgid1` are found, `msgid1` is~~
 3313 | ~~returned.~~

3314 | `msgid2`

3315 | — ~~a NULL terminated string to be returned if the value of `n` is not 1 and no message catalogs are found.~~

3316 | `n`

3317 | — ~~determines which plural form is returned, in a language and message catalog dependent way.~~

Return Value

3318 | ~~On success of a `msgid1` query, the translated NULL terminated string is returned. On error, the original `msgid1` or~~
 3319 | ~~`msgid2` is returned, according to `n`.~~

Errors

3320 | ~~`dngettext` will not modify the `errno` global variable.~~

3321 | `dngettext` shall be equivalent to a call to

3322 | `dcngettext(domainname, msgid1, msgid2, n, LC_MESSAGES)`

3323 | See `dgettext` for more information.

See Also

3324 | gettext (baselib-gettext.html), dgettext, ngettext, ~~dngettext~~, dcgettext, dcgettext, textdomain, bindtextdomain,
 3325 | bind_textdomain_codeset

err

Name

3326 | err — display formatted error messages

Synopsis

3327 | #include <err.h>
 3328 | void **err**(int *eval*, const char **fmt* ...);

Description

3329 | The ~~err displays~~function shall display a formatted error message on the standard error output. The stream. First, ~~err~~
 3330 | shall write the last component of the program name, a colon character, and a space ~~are output~~character. If *fmt* is
 3331 | non-NULL, it shall be used as a format string for the printf family of functions, and ~~err~~ shall write the formatted
 3332 | ~~error message, a colon character, and a space are output. The~~. Finally, the error message string affiliated with the
 3333 | current value of the global variable *errno* is output. The output is shall be written, followed by a newline character.
 3334 | The ~~err does~~function shall not return, but ~~exit~~the program shall terminate with the exit value of *eval*.

See Also

3335 | error, errx

Return Value

3336 | None.

Errors

3337 | None.

error

Name

3338 `error` — print error message

Synopsis

3339 `void error(int exitstatus, int errnum, const char *format ...);`

Description

3340 `error` ~~prints~~shall print a message to standard error.

3341 `error` ~~builds~~shall build the message from the following elements in their specified order:

- 3342 1. the program name. If the application has provided a function named `error_print_progname`, `error` ~~calls~~shall
- 3343 call this to supply the program name; otherwise, `error` uses the content of the global variable `program_name`.
- 3344 2. the colon and space characters, then the result of using the printf-style *format* and the optional arguments.
- 3345 3. if *errnum* is nonzero, `error` ~~adds~~shall add the colon and space characters, then the result of
- 3346 `strerror(errnum)`.
- 3347 4. a newline.

3348 If *exitstatus* is nonzero, `error` ~~calls~~shall call `exit(exitstatus)`.

See Also

3349 `err`, `errx`

errx

Name

3350 | **errx** — formatdisplay formatted error messagesmessage and exit

Synopsis

3351 | #include <err.h>
 3352 | void **errx**(int *eval*, const char **fmt* ...);

Description

3353 | The **errx** displaysfunction shall display a formatted error message on the standard error outputstream. The last
 3354 | component of the program name, a colon character, and a space areshall be output. If *fmt* is non-NULL, it shall be used
 3355 | as the format string for the printf family of functions, and the formatted error message, a colon character, and a
 3356 | space areshall be output. The output. The output is shall be followed by a newline character.
 3357 | **errx** does not return, but **exit**shall exit with the value of *eval*.

Return Value

3358 | None.

Errors

3359 | None.

See Also

3360 | error, err

fcntl

Name

3361 fcntl — file control

Description

3362 fcntl is as specified in the *Single UNIX Specification, Version 3* ISO POSIX (2003), but with differences as listed
3363 below.

Implementation may set O_LARGEFILE

3364 According to the *Single UNIX Specification*, only an application sets fcntl flags, for example *O_LARGEFILE*.
3365 However, this specification also allows implementations an implementation to set *O_LARGEFILE* in a the case in
3366 which where the system default behavior matches the *O_LARGEFILE* behavior.¹ Or in other words, for example if
3367 sizeof(off_t) is 8. Thus, calling fcntl with the *F_GETFL* command may return *O_LARGEFILE* as well as flags
3368 explicitly set by the application.
3369

Notes

3370 1. For example, if *off_t* is 64 bits.
3371

fflush_unlocked

Name

3372 fflush_unlocked — non thread safe fflush

Description

3373 fflush_unlocked is the same as fflush except that it need not be thread safe. That is, it may only be invoked in the
3374 ways which are legal for getc_unlocked.

fgetwc_unlocked

Name

3375 fgetwc_unlocked — non thread safe fgetwc

Description

3376 fgetwc_unlocked is the same as fgetwc except that it need not be thread safe. That is, it may only be invoked in the
3377 ways which are legal for getc_unlocked.

flock

Name

3378 `flock` — apply or remove an advisory lock on an open file

Synopsis

3379 `int flock(int fd, int operation);`

Description

3380 `flock` applies or removes an advisory lock on the open file *fd*. Valid *operation* types are:

3381 `LOCK_SH`

3382 Shared lock. More than one process may hold a shared lock for a given file at a given time.

3383 `LOCK_EX`

3384 Exclusive lock. Only one process may hold an exclusive lock for a given file at a given time.

3385 `LOCK_UN`

3386 Unlock.

3387 `LOCK_NB`

3388 Don't block when locking. May be specified (by *oring*) along with one of the other operations.

3389 A single file may not simultaneously have both shared and exclusive locks.

Return Value

3390 On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

3391 `EWOULDBLOCK`

3392 The file is locked and the `LOCK_NB` flag was selected.

fopen

Name

3393 fopen — open a file

Description

3394 fopen is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

3395 May return ENODEV in place of ENXIO

3396 Where the ~~Single UNIX Specification~~ ISO POSIX (2003) specifies an ENXIO return, the implementation may return
 3397 either ENXIO or ENODEV. Implementations are ~~encouraged~~ encouraged to return ENXIO. ⁺

3398 Notes

3399 ~~1-~~ As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a
 3400 kernel patch would be accepted if submitted.

freopen

Name

3401 freopen — open a file

Description

3402 freopen is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

3403 May return ENODEV in place of ENXIO

3404 Where the ~~Single UNIX Specification~~ ISO POSIX (2003) specifies an ENXIO return, the implementation may return
 3405 either ENXIO or ENODEV. Implementations are ~~encouraged~~ encouraged to return ENXIO. ⁺

3406 Notes

3407 ~~1-~~ As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a
 3408 kernel patch would be accepted if submitted.

getdomainname

Name

3409 | getdomainname — get NIS domain name (DEPRECATED).

Synopsis

```
3410 | #include <unistd.h>
3411 |
3412 | extern int getdomainname (char * name , size_t namelen );
```

Description

3413 | ~~If NIS is in use, provide the NIS domain name. Note that this is not the same as the domain name which provides the~~
 3414 | ~~domain portion of a fully qualified domain name (for example, in DNS). If NIS is not in use, provide the string~~
 3415 | ~~"(none)".~~

3416 | ~~If the string which is provided is strictly less than namelen characters in length, getdomainname places it in the array~~
 3417 | ~~pointed to by name followed by a terminating null character. If not, getdomainname may either truncate it to namelen~~
 3418 | ~~characters and place it in name (without a terminating null character), or may fail with EINVAL.~~

Return Value

3419 | ~~getdomainname returns 0 if successful; -1 if not (in which case errno is set to indicate the error).~~

If the Network Information System (NIS) is in use, `getdomainname` shall copy the NIS domain name to the supplied buffer identified by *name*, with maximum length *name_len*. If the NIS domain name is not currently set, `getdomainname` shall copy the string "(none)" to the *name*. If *name_len* is less the length of the string to be copied, `getdomainname` may either truncate the string to *name_len* characters and place it in *name* (without a terminating null character), or may fail with `EINVAL`.

Note that the NIS domain name is not the same as the domain portion of a fully qualified domain name (for example, in DNS).

Return Value

On success, `getdomainname` shall return 0. Otherwise, it shall return -1 and set `errno` to indicate the error).

Errors

`EINVAL`

name was a null pointer.

`EINVAL`

The buffer identified by *name* and *name_len* is of insufficient size to store the NIS domain name string, and the implementation considers this an error.

Future Directions

The LSB does not include other NIS interfaces, and a future version of this specification may deprecate this interface. Application developers should avoid using this interface where possible.

gethostbyname_r

Name

`gethostbyname_r` — find network host database entry matching host name (DEPRECATED)

Synopsis

```
extern int gethostbyname_r(__const char *__restrict __name, (struct hostent *__restrict
__result_buf), char *__restrict __buf, size_t __buflen, (struct hostent **__restrict
__result), int *__restrict __h_errnop);
```

Description

The `gethostbyname_r` function is deprecated; applications should call `getaddrinfo` instead.

`gethostbyname_r` is a reentrant version of `gethostbyname` that searches the network host database for a host name match.

getloadavg

Name

3442 `getloadavg` — get system load averages

Synopsis

```
3443 #include <stdlib.h>
3444 int getloadavg(double loadavg[], int nelem);
```

Description

3445 `getloadavg` returns the number of processes in the system run queue averaged over various periods of time. Up to
 3446 *nelem* samples are retrieved and assigned to successive elements of *loadavg*[]. The system imposes a maximum of
 3447 3 samples, representing averages over the last 1, 5, and 15 minutes, respectively.

getopt

Name

3448 `getopt` — parse command line options

Synopsis

```
3449 #include <unistd.h>
3450 int getopt(int argc, char * const argv[], const char *optstringoptstring);
3451 extern char *optarg;
```

```
3452 extern int optind, opterr, optopt;
```

Description

3453 The `getopt` function shall parse command line arguments. ~~GNU and POSIX specifications for this function~~
 3454 ~~vary~~ as described in ISO POSIX (2003), with the following ~~are~~ exceptions, where LSB and POSIX specifications
 3455 vary. LSB systems shall implement the GNU modified behaviors described below.

Argument Ordering

3457 The `getopt` function can process command line arguments referenced by *argv* in one of three ways:

PERMUTE

3459 the order of arguments in *argv* is altered so that all options (and their arguments) are moved in front of all of the
 3460 operands. This is the default behavior.

3461 This behavior has undefined results if *argv* is not modifiable. This is to support historic behavior predating
 3462 the use of `const` and ISO C (1999). The function prototype was aligned with ISO POSIX (2003) despite the
 3463 fact that it modifies *argv*, and the library maintainers are unwilling to change this.

REQUIRE_ORDER

3465 The arguments in *argv* are processed in exactly the order given, and option processing stops when the first
 3466 non-option argument is reached, or when the element of *argv* is `"--"`. This ordering can be enforced either by
 3467 setting the environment variable `POSIXLY_CORRECT`, or by setting the first character of *optstring* to `+`.

RETURN_IN_ORDER

3469 The order of arguments is not altered, and all arguments are processed. Non-option arguments (operands) are
 3470 handled as if they were the argument to an option with the value 1 (`'001'`). This ordering is selected by setting the
 3471 first character of *optstring* to `-`;

Option Characteristics

3473 ~~GNU~~LSB specifies that:

- 3474 • an element of *argv* that starts with `"-"` (and is not exactly `"-"` or `"--"`) is an option element.
- 3475 • characters of an option element, aside from the initial `"-"`, are option characters.

3476 POSIX specifies that:

- 3477 • applications using `getopt` shall obey the following syntax guidelines:
 - 3478 • option name is a single alphanumeric character from the portable character set
 - 3479 • option is preceded by the `"'"'` delimiter character
 - 3480 • options without option-arguments should be accepted when grouped behind one `"'"'` delimiter
 - 3481 • each option and option-argument is a separate argument
 - 3482 • option-arguments are not optional
 - 3483 • all options should precede operands on the command line
 - 3484 • the argument `"--"` is accepted as a delimiter indicating the end of options and the consideration of subsequent
 - 3485 arguments, if any, as operands

- historical implementations of `getopt` support other characters as options as an allowed extension, but applications that use extensions are not maximally portable.
- support for multi-byte option characters is only possible when such characters can be represented as type `int`.
- applications that call any utility with a first operand starting with `"--"` should usually specify `"--"` to mark the end of the options. Standard utilities that do not support this guideline indicate that fact in the `OPTIONS` section of the utility description.

Extensions

GNU *LSB* specifies that:

- if a character is followed by two colons, the option takes an optional argument; if there is text in the current *argv* element, it is returned in *optarg*, otherwise *optarg* is set to 0.
- if *optstring* contains *w* followed by a semi-colon (;), then `-w foo` is treated as the long option `--foo`. (Not available with libraries before *GNU libc 2*.)

See `getopt_long` for a description of long options.

- The first character of *optstring* shall modify the behavior of `getopt` as follows:
 - if the first character is '+', then `REQUIRE_ORDER` processing shall be in effect (see above)
 - if the first character is '-', then `RETURN_IN_ORDER` processing shall be in effect (see above)
 - if the first character is ':', then `getopt` shall return ':' instead of '?' to indicate a missing option argument, and shall not print any diagnostic message to `stderr`.

POSIX specifies that:

- the `-w` option is reserved for implementation extensions.

Return Values

GNU specifies the following `getopt` return values:

- the next option character is returned, if found successfully.
- `":"` is returned if a parameter is missing for one of the options.
- `"?"` is returned if an unknown option character is encountered.
- `-1` is returned for the end of the option list.

LSB specifies the following additional `getopt` return values:

- `'\001'` is returned if `RETURN_IN_ORDER` argument ordering is in effect, and the next argument is an operand, not an option. The argument is available in *optarg*.

Any other return value has the same meaning as for *POSIX*.

POSIX specifies the following `getopt` return values:

- the next option character is returned, if found successfully.
- `":":'` is returned if a parameter is missing for one of the options and the first character of *optstring* is `":"`. *optstring* is `':'`.

- "?" is returned if an unknown option character not in `optstring` is encountered, or if `getopt` detects a missing argument and the first character of `optstring` is not "-".
- -1 is returned for the end of the option list.

Environment Variables

*GNU*LSB specifies that:

- if the variable `POSIXLY_CORRECT` is set, option processing stops as soon as a non-option argument is encountered.
- ~~if `POSIXLY_CORRECT`~~ the variable `_[PID]_GNU_nonoption_argv_flags_` (where `[PID]` is set, *GNU* ~~`getopt` conforms~~ the process ID for the current process), contains a space separated list of arguments that should not be treated as arguments even though they appear to *ISO/IEC 9945:2003 Portable Operating System (POSIX)* and *The Single UNIX® Specification (SUS) V3*. be so.

~~the variable `_[PID]_GNU_nonoption_argv_flags_`~~ **Rationale**

This was used by bash 2.0 to communicate to *GNU*libc which arguments resulted from wildcard expansion and so should not be considered as options. This behavior was removed in bash version 2.01, but the support remains in *GNU*libc.

This behavior is DEPRECATED in this version of the LSB; future revisions of this specification may not include this requirement.

getopt_long

Name

`getopt_long` — parse command line options

Synopsis

```
#define _GNU_SOURCE
#include <getopt.h>
int getopt_long(int argc, char * const argv[], const char *opstring, (const struct option
*longopts), int *longindex);
```

Description

`getopt_long` works like `getopt` except that it also accepts long options, started out by two dashes. Long option names may be abbreviated if the abbreviation is unique or is an exact match for some defined option. A long option may take a parameter, of the form `--arg=param` or `--arg param`.

`longopts` is a pointer to the first element of an array of struct option declared in `getopt.h` as:

```
struct option {
    const char *name;
    int *flag;
    int has_arg;
    int *flag;
    int val;
```

```
3551     };
```

Return Value

```
3552 getopt_long returns the option character if the option was found successfully, or ":" if there was a missing
3553 parameter for one of the options, or "?" for an unknown option character, or -1 for the end of the option list.
```

```
3554 getopt_long also returns the option character when a short option is recognized. For a long option, they return val if
3555 flag is NULL, and 0 otherwise. Error and -1 returns are the same as for getopt, plus "?" for an ambiguous match or an
3556 extraneous parameter.
```

```
3557 The fields in this structure have the following meaning:
```

```
3558     name
```

```
3559         The name of the long option.
```

```
3560     has_arg
```

```
3561         One of:
```

```
         argument (or 0) if the option does not take an argument,
         uired_argument (or 1) if the option requires an argument, or
3562         ional_argument (or 2) if the option takes an optional argument.
```

```
3563     flag
```

```
3564         specifies how results are returned for a long option. If flag is NULL, then getopt_long shall return val. (For
3565         example, the calling program may set val to the equivalent short option character.) Otherwise, getopt_long
3566         returns 0, and flag shall point to a variable which shall be set to val if the option is found, but left unchanged
3567         if the option is not found.
```

```
3568     val
```

```
3569         The value to return, or to load into the variable pointed to by flag.
```

Return Value

```
3570 getopt_long returns the option character if a short option was found successfully, or ":" if there was a missing
3571 parameter for one of the options, or "?" for an unknown option character, or -1 for the end of the option list.
```

```
3572 For a long option, getopt_long returns val if flag is NULL, and 0 otherwise. Error and -1 returns are the same as
3573 for getopt, plus "?" for an ambiguous match or an extraneous parameter.
```

getopt_long_only

Name

```
3574 getopt_long_only — parse command line options
```

Synopsis

```
3575 #define _GNU_SOURCE
```

```

3576 #include <getopt.h>
3577 | int getopt_long_only(int argc, char * const argv[], const char *optstring, (const
3578 struct option *longopts), int *longindex);

```

Description

3579 `getopt_long_only` is like `getopt_long`, but "-" as well as "--" can indicate a long option. If an option that starts
 3580 with "-" (not "--") doesn't match a long option, but does match a short option, it is parsed as a short option instead.

Return Value

3581 `getopt_long_only` returns the option character if the option was found successfully, or ":" if there was a missing
 3582 parameter for one of the options, or "?" for an unknown option character, or -1 for the end of the option list.

3583 `getopt_long_only` also returns the option character when a short option is recognized. For a long option, they
 3584 return val if flag is NULL, and 0 otherwise. Error and -1 returns are the same as for `getopt`, plus "?" for an ambiguous
 3585 match or an extraneous parameter.

gettext

Name

3586 | `gettext` — perform lookup in search message catalog catalogs for the current `LC_MESSAGES` locale a string

Synopsis

```
3587 | #include <libintl.h>
3588 | extern char *gettext(const char *msgid);
```

Description

3589 | `gettext` attempts to retrieve a target string based on the specified key from `msgid` within the context of a specific
 3590 | domain and the current locale.

3591 | The `LANGUAGE` environment variable is examined first to determine the message catalogs to be used. `LANGUAGE` is a
 3592 | list of locale names separated by ":" character. If `LANGUAGE` is defined, each locale name is tried in the specified order
 3593 | and if a message catalog containing the requested message is found, the message is returned. If `LANGUAGE` is defined
 3594 | but failed to locate a message catalog, the `msgid` string is returned. If `LANGUAGE` is not defined, the `LC_ALL`, `LC_XXX`,
 3595 | and `LANG` environment variables are examined to locate the message catalog, following the convention used by the
 3596 | `setlocale` function.

3597 | The pathname used to locate the message catalog is `dirname/locale/category/domainname.mo`, where `dirname`
 3598 | is the directory specified by the `bindtextdomain` function, `locale` is a locale name determined by the definition of
 3599 | environment variables, and `category` is `LC_MESSAGES`.

3600 | If the `LC_MESSAGES` locale category of the current locale is the standard C locale or the standard POSIX locale,
 3601 | `gettext` returns `msgid` without looking in any message catalog.

Parameters

3602 | `msgid`
 3603 | — A `NULL` terminated string to be matched in the catalogue with respect to a specific domain and the current locale.

Return Value

3604 | If the function query above succeeds with `msgid`, then a translated `NULL` terminated string is returned. If the search
 3605 | fails, then the original `msgid` is returned. The length of the string returned is undetermined until the function is called.

Errors

3606 | `gettext` does not modify the global variable `errno`.

See Also

3607 | `gettext` (base/lib-gettext.html), `dgettext`, `ngettext`, `dngettext`, `degettext`, `dengettext`, `textdomain`, `bindtextdomain`,
 3608 | `bind_textdomain_codeset`

3609 The `gettext` function shall search the currently selected message catalogs for a string identified by the string *msgid*.
 3610 If a string is located, that string shall be returned.
 3611 The `gettext` function is equivalent to `dcgettext(NULL, msgid, LC_MESSAGES)`.

Return Value

3612 If a string is found in the currently selected message catalogs for *msgid*, then a pointer to that string shall be returned.
 3613 Otherwise, a pointer to *msgid* shall be returned.
 3614 Applications shall not modify the string returned by `gettext`.

Errors

3615 None.
 3616 The `gettext` function shall not modify `errno`.

See Also

3617 `dgettext`, `ngettext`, `dngettext`, `dcgettext`, `dcngettext`, `textdomain`, `bindtextdomain`, `bind_textdomain_codeset`

getutent

Name

3618 `getutent` — access `utmp` file user accounting database entries

Synopsis

3619 `#include <utmp.h>`
 3620 `struct utmp *getutent(void);`

Description

3621 `getutent` reads a line from the current file position in the `utmp` file. It returns a pointer to a structure containing the
 3622 fields of the line.

Return Value

3623 `getutent` returns a pointer to a static `struct utmp`.

Errors

3624 On error, `(struct utmp*)0` is returned.

Files

3625 `/var/run/utmp` database of currently logged in users
 3626 `/var/log/wtmp` database of past user logins

3627 The `getutent` function shall read the next entry from the user accounting database.

Return Value

3628 Upon successful completion, `getutent` shall return a pointer to a `utmp` structure containing a copy of the requested
 3629 entry in the user accounting database. Otherwise, a null pointer shall be returned. The return value may point to a static
 3630 area which is overwritten by a subsequent call to `getutent`.

Errors

3631 None defined.

getutent_r

Name

3632 `getutent_r` — access ~~utmp file~~ user accounting database entries

Synopsis

3633 ~~extern int~~ **`getutent_r`**(~~struct utmp *~~*buffer*), (*struct utmp ***~~result~~);

Description

3634 The `getutent_r` function is a **reentrant version of the `getutent` ~~utmp file handler~~**function. On entry, *buffer*
 3635 should point to a user supplied buffer to which the next entry in the database will be copied, and *result* should point
 3636 to a location where the result will be stored.

Return Value

3637 On success, `getutent_r` shall return 0 and set the location referenced by *result* to a pointer to *buffer*.
 3638 Otherwise, `getutent_r` shall return -1 and set the location referenced by *result* to NULL.

glob64

Name

3639 glob64 — find pathnames matching a pattern (Large File Support)

Synopsis

```
3640 #include <glob.h>
3641 int glob64(const char *pattern, int flags, int (*errfunc) (const char *, int), glob64_t
3642 *pglob);
```

Description

3643 The glob64 searches for all-function is a large-file version of the pathnames matching pattern according to the
 3644 rules used by the shell. (See glob(7).) defined in ISO POSIX (2003). It shall search for pathnames matching
 3645 pattern according to the rules used by the shell, /bin/sh. No tilde expansion or parameter substitution is done; if
 3646 you want these, use wordexp(3).

3647 The results of a glob64 call are stored in the structure pointed to by pglob, which is a glob64_t declared in
 3648 glob.h and includes with the following elements defined by POSIX.2 (more may be present as an
 3649 extension): members:

```
3650 typedef struct
3651 {
3652     size_t gl_pathc;
3653     char **gl_pathv;
3654     size_t gl_offs;
3655     int gl_flags;
3656     void (*gl_closedir) (void *);
3657     struct dirent64 *(*gl_readdir64) (void *);
3658     void *(*gl_opendir) (const char *);
3659     int (*gl_lstat) (const char *, struct stat *);
3660     int (*gl_stat) (const char *, struct stat *);
3661 }
```

```

3662 glob64 is a 64-bit version of _t;
3663 Structure members with the same name as corresponding members of a glob_t as defined in ISO POSIX (2003) shall
3664 have the same purpose.
3665 Other members are defined as follows:
3666 gl_flags
3667     reserved for internal use
3668 gl_closedir
3669     pointer to a function capable of closing a directory opened by gl_opendir
3670 gl_readdir64
3671     pointer to a function capable of reading entries in a large directory
3672 gl_opendir
3673     pointer to a function capable of opening a large directory
3674 gl_stat
3675     pointer to a function capable of returning file status for a large file
3676 gl_lstat
3677     pointer to a function capable of returning file status information for a large file or symbolic link
3678 A large file or large directory is one with a size which cannot be represented by a variable of type off_t.

```

Return Value

```

3679 On success, 0 is returned. Other possible returns are:
3680 GLOB_NOSPACE
3681     out of memory
3682 GLOB_ABORTED
3683     read error
3684 GLOB_NOMATCH
3685     no match found

```

globfree64

Name

3686 globfree64 — free memory from glob64() (Large File Support)

Synopsis

```
3687 #include <glob.h>
3688 void globfree64(glob64_t *pglob);
```

Description

3689 globfree64 frees the dynamically allocated storage from an earlier call to glob64.
3690 globfree64 is a 64-bit version of globfree.

initgroups

Name

3691 initgroups — initialize the supplementary group access list

Synopsis

```
3692 #include <grp.h>
```

```

3693 #include <sys/types.h>
3694 int initgroups(const char *user, gid_t group);

```

Description

```

3695 initgroups initializes the group access list. If the calling process has appropriate privilege, the initgroups function shall
3696 initialize the Supplementary Group IDs for the current process by reading the group database and using all groups of
3697 which user is a member. The additional group group is also added to the list.

```

Return Value

```

3698 On success, 0 is returned. On error, -1 is returned and the global variable errno is set appropriately.

```

Errors

```

3699 EPERM
3700     The calling process does not have sufficient privileges.
3701 ENOMEM
3702     Insufficient memory to allocate group information structure.

```

See Also

```

3703 setgroups

```

ioctl

Name

3704 `ioctl` — control device

Synopsis

```
3705 #include <sys/ioctl.h>
3706 int ioctl (int d , int request , ... );
```

Description

3707 The `ioctl()` function shall manipulate the underlying device parameters of special files. *d* shall be an open file
 3708 descriptor referring to a special file. The `ioctl()` function shall take three parameters; the type and value of the third
 3709 parameter is dependent on the device and *request*.

3710 ~~An application may~~ Conforming LSB applications shall not call `ioctl` except for in situations explicitly stated in this
 3711 specification.

Return Value

3712 On success, 0 is returned. An `ioctl` may use the return value as an output parameter and return a non-negative value
 3713 on success. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

3714 EBADF

3715 *d* is not a valid descriptor.

3716 EFAULT

3717 The third parameter references an inaccessible memory area.

3718 ENOTTY

3719 *d* is not associated with a character special device.

3720 ENOTTY

3721 The specified request does not apply to the kind of object that *d* references.

3722 EINVAL

3723 *request* or the third parameter is not valid.

sockio

Name

3724 sockio — socket ioctl commands

Synopsis

3725 #include <sys/socket.h>
3726 #include <net/if.h>

```

3727 #include <netinet/in.h>
3728 int ioctl(int sockfd, int request, char *argp);

```

Description

3729 Socket `ioctl` commands are a subset of the `ioctl` calls, which can perform a variety of functions on sockets.

3730 `sockfd` shall contain the value of a file descriptor that was created with the `socket` or `accept` calls.

3731 Socket `ioctl` commands apply to the underlying network interfaces, and affect the entire system, not just the file
 3732 descriptor used to issue the `ioctl`.

3733 The following ~~`ioctl`~~ values for `request` are provided/accepted:

3734 SIOCGIFCONF

3735 Gets the interface configuration list for the system. ¹~~`argp`~~

3736 SIOCGIFCONF is a pointer similar to the `if_nameindex` family found in the ISO POSIX (2003) or the
 3737 `getifaddrs` family found in BSD derived systems.

3738 `argp` shall point to a `ifconf` structure, as described in `<net/if.h>`. Before calling, the caller shall allocate-set
 3739 the `ifc_ifcu.ifcu_req` field to point to an array of `ifreq` structures, and set `ifc_len` to the size in bytes of
 3740 this allocated array (in bytes). Upon return, `ifc_len` will contain the amount/size in bytes of the array which
 3741 was actually used (again, in bytes). If it is the same as the length upon calling, the caller should assume that the
 3742 array was too small and try again with a larger array.

3743 On success, SIOCGIFCONF can return any nonnegative value. ²

3744 Rationale

3745 Historical UNIX systems disagree on the meaning of the return value.

3746 SIOCGIFFLAGS

3747 Gets the interface flags for the indicated interface. ~~`argp` is a pointer~~ shall point to a `ifreq` structure. Before calling,
 3748 the caller should fill in the `ifr_name` field with the interface name, and upon return, the
 3749 `ifr_ifru.ifru_flags` field is set with the interface flags.

3750 SIOCGIFADDR

3751 Gets the interface address list for the system/given interface. ~~`argp` is a pointer~~ shall point to a `ifreq` structure.
 3752 Before calling, the caller should fill in the `ifr_name` field with the interface name, and upon return, the
 3753 `ifr_ifru.ifru_addr` field is set with the interface address.

3754 SIOCGIFNETMASK

3755 Gets the network mask for the indicated/given interface. ~~`argp` is a pointer~~ shall point to a `ifreq` structure. Before
 3756 calling, the caller should fill in the `ifr_name` field with the interface name, and upon return, the
 3757 `ifr_ifru.ifru_netmask` field is set with the network mask.

3758 FIONREAD

3759 Returns the amount of queued unread data in the receive buffer. ~~Argument is a pointer~~ `argp` shall point to an
 3760 integer where the result is to be placed.

3761 The `sockaddr` structure is as specified in the *Single UNIX Specification*.

Return Value

3762 On success, if *request* is SIOCGIFCONF, a non-negative integer shall be returned. If request is not
 3763 SIOCGIFCONF, on success 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

3764 EBADF

3765 *sockfd* is not a valid descriptor.

3766 EFAULT

3767 *argp* references an inaccessible memory area.

3768 ENOTTY

3769 ~~— *sockfd* is not associated with a character special device.~~

3770 ~~ENOTTY~~

3771 The specified *request* does not apply to the kind of object that the descriptor *sockfd* references.

3772 EINVAL

3773 ~~Either *request* and/or *argp* are not~~ is invalid.

3774 ENOTCONN

3775 The operation is only defined on a connected socket, but the socket wasn't connected.

Notes

3777 1. ~~SIOCGIFCONF is similar to the `if_nameindex` family found in the *Single UNIX Specification, Version 3* or the~~
 3778 ~~`getifaddrs` family found in BSD.~~

3779 2. ~~Historical UNIX systems disagree on the meaning of the return value.~~

iswetype

Name

3780 ~~iswetype~~ — wide character classification

Synopsis

3781 ~~#include <wctype.h>~~
3782 ~~int iswetype(wint_t wc, wctype_t desc);~~

Description

3783 ~~iswetype tests *wc* to determine if it is a wide character whose property is designated by the character class *desc*.~~
3784 ~~*desc* shall be a character property descriptor returned by the *wctype* function.~~

Return Value

3785 ~~If *wc* belongs to the character class *desc*, a nonzero value is returned. Otherwise, 0 is returned.~~
3786 ~~Note that if *wc* is WEOF, 0 is returned.~~

Notes

3787 ~~The behavior of *iswetype* depends on the *LC_CTYPE* category of the current locale.~~

kill

Name

3788 kill — send a signal

Synopsis

```
3789 | #include <signal.h>
3790 | int kill(pid_t pid, int sig);
```

Description

3791 | kill is as specified in the *Single UNIX Specification, Version 2 ISO POSIX (2003)*, but with differences as listed
3792 | below.

Process ID -1 doesn't affect calling process

3794 | If *pid* is specified as -1, *sig* shall not be sent to the calling process. [†]Other than this, the rules in the *Single UNIX*
3795 | *Specification, Version 2 ISO POSIX (2003)* apply.

Notes

3797 | [†] — **Rationale**

3798 | This was a deliberate Linus decision after an unpopular experiment in including the calling process in the 2.5.1
3799 | kernel. See "What does it mean to signal everybody?", Linux Weekly News, 20 December 2001,
3800 | <http://lwn.net/2001/1220/kernel.php3>

mbsnrtowcs

Name

3801 mbsnrtowcs — convert a multibyte string to a wide character string

Synopsis

```
3802 | #include <wchar.h>
3803 | size_t mbsnrtowcs(wchar_t *dest, const char **src, size_t nms, size_t len, mbstate_t *ps);
```

Description

3804 | mbsnrtowcs is like mbsrtowcs, except that the number of bytes to be converted, starting at *src*, is limited to *nms*.

3805 | If *dest* is not a NULL pointer, mbsnrtowcs converts at most *nms* bytes from the multibyte string *src* to a
3806 | wide-character string starting at *dest*. At most, *len* wide characters are written to *dest*. The state *ps* is updated.

3807 | The conversion is effectively performed by repeatedly calling:

3808

3809 `mbrtowc(dest, *src, n, ps)`

3810 where *n* is some positive number, as long as this call succeeds, and then incrementing *dest* by one and *src* by the
3811 number of bytes consumed.

3812 The conversion can stop for three reasons:

- 3813 • An invalid multibyte sequence has been encountered. In this case *src* is left pointing to the invalid multibyte
3814 sequence, `(size_t)(-1)` is returned, and `errno` is set to `EILSEQ`.
- 3815 • The *nms* limit forces a stop, or *len* non-`L'\0'` wide characters have been stored at *dest*. In this case, *src* is left
3816 pointing to the next multibyte sequence to be converted, and the number of wide characters written to *dest* is
3817 returned.
- 3818 • The multibyte string has been completely converted, including the terminating `\0'` (which has the side effect of
3819 bringing back *ps* to the initial state). In this case, *src* is set to `NULL`, and the number of wide characters written to
3820 *dest*, excluding the terminating `L'\0'` character, is returned.

3821 If *dest* is `NULL`, *len* is ignored, and the conversion proceeds as above, except that the converted wide characters are
3822 not written out to memory, and that no destination length limit exists.

3823 In both of the above cases, if *ps* is a `NULL` pointer, a static anonymous state only known to `mbnrtowcs` is used
3824 instead.

3825 The programmer shall ensure that there is room for at least *len* wide characters at *dest*.

Return Value

3826 `mbnrtowcs` returns the number of wide characters that make up the converted part of the wide character string, not
3827 including the terminating null wide character. If an invalid multibyte sequence was encountered, `(size_t)(-1)` is
3828 returned, and the global variable `errno` is set to `EILSEQ`.

Notes

3829 The behavior of `mbnrtowcs` depends on the `LC_CTYPE` category of the current locale.

3830 Passing `NULL` as *ps* is not multi-thread safe.

memmem

Name

3831 `memmem` — locate a substring

Synopsis

3832 `#define _GNU_SOURCE`

```

3833 #include <string.h>
3834 void *memmem(const void *haystack, size_t haystacklen, const void *needle, size_t
3835 needlelen);

```

Description

3836 memmem finds the start of the first occurrence of the `substring` byte array referenced by `needle` of length `needlelen`
 3837 in the memory area `haystack` of length `haystacklen`.

Return Value

3838 memmem returns a pointer to the beginning of the `substring` byte array, or NULL if the `substring` byte array is not found.

Notes

3839 Earlier versions of the C library (prior to glibc 2.1) contained a memmem ~~was broken in Linux libraries up to~~ with
 3840 various problems, ~~and including libc 5.0.9; there the needle and haystack arguments were interchanged, and a~~
 3841 ~~pointer to the end of the first occurrence of needle was returned. Since libc 5.0.9 is still widely used, application~~
 3842 ~~developers should treat this is a dangerous function to use.~~

3843 ~~Both old and new libc's have the bug that if needle is empty, haystack - 1 is returned (instead of haystack). And~~
 3844 ~~glibc 2.0 makes it worse, returning a pointer to the last byte of haystack. This is fixed in glibc 2.1 with care.~~

memrchr

Name

3845 memrchr — scan memory for a character

Synopsis

```
3846 #include <string.h>
3847 void *memrchr(const void *s, int c, size_t n);
```

Description

3848 The `memrchr` returns a pointer to the function shall locate the last occurrence of `c` (converted to an unsigned
 3849 char) in the first initial `n` characters bytes (each interpreted as an unsigned char) of the string represented object pointed
 3850 to by `s`.

Return Value

3851 The `memrchr` shall return a pointer to the located byte, or a null pointer if the byte does not occur in the object.

Errors

3852 No errors are defined.

See Also

3853 memchr

ngettext

Name

3854 | `ngettext` — perform lookup in Search message catalog catalogs for the current LC_MESSAGES locale plural string

Synopsis

```
3855 | #include <libintl.h>
3856 | extern char *ngettext(const char *msgid1, const char *msgid2, unsigned long int n);
```

Description

3857 | `ngettext` is the plural version of `gettext`, which searches for the message string using the `msgid1` arguments as
 3858 | the key, using the argument `n` to determine the plural form. If no message catalogs containing a translation for
 3859 | `msgid1` are found, `msgid1` is returned if `n == 1`, otherwise, `msgid2` is returned. (See `gettext` for more details.)

Parameters

3860 | `msgid1`
 3861 | — A NULL terminated string to be matched in the catalogue with respect to a specific domain and the current locale.
 3862 | If no message catalogs are found, `msgid1` is returned if `n == 1`.

3863 | `msgid2`
 3864 | — A NULL terminated string to be returned if no message catalogs are found and `n != 1`.

3865 | `n`
 3866 | — Determines in which plural form a message string is returned, in a language and message catalog dependent way.

Return

3867 | If the function query above succeeds with `msgid1`, then a translated NULL terminated string is returned. If the search
 3868 | fails, then the original `msgid1` or `msgid2` is returned, according to `n`.

Errors

3869 | `ngettext` will not modify the `errno` global variable.

```
3870 | char *ngettext(const char *msgid1, const char *msgid2, unsigned long int n);
```

Description

3871 | The `ngettext` function shall search the currently selected message catalogs for a string matching the singular string
 3872 | `msgid1`. If a string is located, and if `n` is 1, that string shall be returned. If `n` is not 1, a pluralized version (dependant
 3873 | on `n`) of the string shall be returned.

3874 | The `ngettext` function is equivalent to `dcngettext(NULL, msgid1, msgid2, n, LC_MESSAGES)`.

Return Value

3875 If a string is found in the currently selected message catalogs for *msgid1*, then if *n* is 1 a pointer to the located string
 3876 shall be returned. If *n* is not 1, a pointer to an appropriately pluralized version of the string shall be returned. If no
 3877 message could be found in the currently selected message catalogs, then if *n* is 1, a pointer to *msgid1* shall be returned,
 3878 otherwise a pointer to *msgid2* shall be returned.
 3879 Applications shall not modify the string returned by `ngettext`.

Errors

3880 None.
 3881 The `ngettext` function shall not modify `errno`.

See Also

3882 `gettext` (baselib-gettext.html), `dgettext`, `ngettext`, `dngettext`, `dcgettext`, `dcngettext`, `textdomain`, `bindtextdomain`,
 3883 `bind_textdomain_codeset`

obstack_free

Name

3884 `obstack_free` — free an object in the obstack

Synopsis

```
3885 #include <obstack.h>
3886 void obstack_free((struct obstack *obstack), void *block);
```

Description

3887 `obstack_free` frees an object in the obstack.

Future Directions

3888 Future versions of this specification may not include support for this interface.

open

Name

3889 `open` — open a file

Description

Synopsis

```
3890 #include <sys/stat.h>
```

```

3891 #include <fcntl.h>
3892 int open-is(const char *path, int oflag, ...);

```

Description

3893 The open function shall behave as specified in *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and*
 3894 *The Single UNIX® Specification (SUS) V3*), but except with differences as listed below.

3895 May return ENODEV in place of ENXIO

3896 Where *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)*
 3897 *V3* specifies an ENXIO return, a conforming implementation may return either ENXIO or ENODEV.
 3898 Implementations are encouraged to return ENXIO. ⁺

3899 Notes

3900 1. — Rationale

3901 As of spring 2004, we don't know of anyno Linux kernel patches to switch to ENXIO are known, but we believe it is
 3902 believed that such a kernel patch would be accepted if submitted.

opterr

Name

3903 opterr — external variable used in getopt()

Synopsis

```

3904 extern int opterr;

```

Description

3905 opterr is used as a flag to suppress an error message generated by getopt. When opterr is set to 0, it suppresses
 3906 the error message generated by getopt when that function does not recognize an option character.

optind

Name

3907 `optind` — external variable used in `getopt()`

Synopsis

3908 `extern int optind;`

Description

3909 `optind` holds the current index of the array `argv`, which contains the command line options being parsed by
 3910 `getopt`.

optopt

Name

3911 `optopt` — external variable used in `getopt()`

Synopsis

3912 `extern int optopt;`

Description

3913 `optopt` holds the unknown option character when that option character is not recognized by `getopt`.

pmap_getport

Name

3914 `pmap_getport` — ~~Returns~~ Find the port number on which assigned to a service is waiting for registered with a
 3915 portmapper.

Synopsis

3916 `#include <pmap_clnt.h>`
 3917 ~~extern u_short~~ **`*pmap_getport`**(struct sockaddr_in *address, __const u_long program, __const
 3918 u_long *version, u_int protocol);

Description

3919 The `pmap_getport` ~~returns~~ function shall return the port number on which assigned to a service is waiting for.
 3920 ~~`pmap_getport` is called given the RPC program number *program*, *version*, and the transport *protocol* set to~~
 3921 ~~either IPPROTO_UDP or IPPROTO_TCP. The pre-allocated socket *address* is registered with a returned parameter.~~

Return Value

3922 ~~`pmap_getport` returns 0 if the mapping does not exist or if contact to the remote portmap~~RPC Binding service
 3923 ~~failed~~running on a given target system, using the protocol described in RFC 1833: Binding Protocols for ONC RPC
 3924 Version 2. The `pmap_getport` function shall be called given the RPC program number *program*, the program
 3925 version *version*, and transport protocol *protocol*. Conforming implementations shall support both
 3926 IPPROTO_UDP and IPPROTO_TCP protocols. On entry, *address* shall specify the address of the system on which the
 3927 portmapper to be contacted resides. The value of *address*->*sin_port* shall be ignored, and the standard value for
 3928 the portmapper port shall always be used.

3929 Security and network restrictions may prevent a conforming application from contacting a remote RPC Binding
 3930 Service.

Return Value

3931 On success, the `pmap_getport` function shall return the port number in host byte order of the RPC application
 3932 registered with the remote portmapper. On failure, if either the program was not registered or the remote portmapper
 3933 service could not be reached, the `pmap_getport` function shall return 0. If the remote portmap service could not be
 3934 reached, the status is left in the global variable `rpc_createerr`.

pmap_set

Name

3935 | `pmap_set` — Establishes mapping to machine's `portmap` RPC Bind service.

Synopsis

```
3936 #include <rpc/pmap_clnt.h>
3937 *pmap_set(__const u_long program, __const u_long version, int protocol, u_short port);
```

Description

3938 `pmap_set` establishes a mapping between the triple `[program,version,protocol]` and `port` on the
 3939 | machine's `portmap` RPC Bind service. The value of `protocol` is most likely `IPPROTO_UDP` or `IPPROTO_TCP`.
 3940 Automatically done by `svc_register`.

Return Value

3941 `pmap_set` returns 1 if it succeeds, 0 otherwise.

pmap_unset

Name

3942 | `pmap_unset` — Destroys all mapping between the triple and ports. RPC Binding

Synopsis

```
3943
3944 #include <rpc/rpc.h>
3945
3946 void pmap_unset(u_long prognum, u_long versnum);
```

Description

3947 As a user interface to the `portmap` RPC Bind service, `pmap_unset` destroys all mapping between the triple
 3948 | `[prognum,versnum, *]` and ports on the machine's `portmap` RPC Bind service.

Return Value

3949 `pmap_unset` returns 1 if it succeeds, zero otherwise.

psignal

Name

3950 `psignal` — print signal message

Synopsis

```
3951 #include <signal.h>
3952 void psignal(int sig, const char *s);
3953 extern const char *const sys_siglist[]
```

Description

3954 The `psignal` displaysfunction shall display a message on the `stderr` consistingstream. If *s* is not the null pointer,
 3955 and does not point to an empty string (e.g. `"\0"`), the message shall consist of the string *s*, a colon, a space, and a
 3956 string describing the signal number *sig*; otherwise `psignal` shall display only a message describing the signal
 3957 number *sig*. If *sig* is invalid, the message displayed willshall indicate an unknown signal.

3958 The array `sys_siglist` holds the signal description strings indexed by signal number.

Return Value

3959 `psignal` returns no value.

random_r

Name

3960 `random_r` — generate random number

Synopsis

```
3961 extern int random_r((struct random_data *__restrict __buf), int32_t *__restrict __result);
```

Description

3962 `random_r` is a reentrant version of `random`, which generates a pseudorandom number.

Future Directions

3963 Since this function requires support from other functions not specified in this specification (most notably
 3964 `initstate_r`), a future version of this specification may deprecate this interface.

setbuffer

Name

3965 `setbuffer` — stream buffering operation

Synopsis

```
3966    #include <stdio.h>
3967    void setbuffer(FILE *stream, char *buf, size_t size);
```

Description

3968 `setbuffer` is an alias for the call to `setvbuf`. It works the same, except that the size of the buffer in `setbuffer` is
 3969 up to the caller, rather than being determined by the default `BUFSIZ`.

setdomainname

Name

3970 `setdomainname` — set NIS domain name (DEPRECATED).

Synopsis

```
3971    #include <unistd.h>
3972    |
```



```
3973 | extern int setdomainname (char * name , size_t namelen );
```

Description

3974 If NIS is in use, set the NIS domain name. Note that this is not the same as the domain name which provides the
 3975 domain portion of a fully qualified domain name (for example, in DNS). If NIS is not in use, this function may set the
 3976 domain name anyway, or it may fail.

3977 This call shall fail unless the caller has appropriate privileges.

3978 *namelen* shall be the length of the string pointed to by *name*.

Return Value

3979 On success, **setdomainname** ~~returns~~ shall return **0** ~~if successful~~. Otherwise, it shall return **-1** ~~if not (in which case and~~
 3980 ~~set errno is set to indicate the error).~~

Errors

3981 EPERM

3982 The process did not have sufficient privilege to set the domain name.

3983 EINVAL

3984 *name* is a null pointer.

setgroups

Name

3985 `setgroups` — set list of supplementary group IDs

Synopsis

```
3986 #include <grp.h>
3987 int setgroups(size_t size, const gid_t *list);
```

Description

3988 If the process has appropriate privilege, the `setgroups` sets-function shall set the supplementary group IDs for
 3989 the process. ~~Only the super-user~~current process. *list* shall reference an array of *size* group IDs. A process may use
 3990 ~~this function~~have at most NGROUPS_MAX supplementary group IDs.

Return Value

3991 On successful completion, 0 is returned. On error, -1 is returned and the ~~global variable~~`errno` is set appropriately to
 3992 indicate the error.

Errors

3993 EFAULT

3994 *list* has an invalid address.

3995 EPERM

3996 The ~~user is~~process does not ~~the super-user~~have appropriate privileges.

3997 EINVAL

3998 *size* is greater than NGROUPS ~~(32 for Linux 2.0.32)~~_MAX.

sethostid

Name

3999 `sethostid` — set the unique identifier of the current host

Synopsis

```
4000 #include <unistd.h>
4001 int sethostid(long int hostid);
```

Description

4002 `sethostid` sets a unique 32-bit identifier for the current machine. The 32-bit identifier is intended to be unique
 4003 among all UNIX systems in existence. This normally resembles the Internet address for the local machine as returned
 4004 by `gethostbyname(3)`, and thus usually never needs to be set.

4005 The `sethostid` call is restricted to the superuser.

4006 *hostid* is stored in the file `/etc/hostid`.

Return Value

4007 `gethostid` returns the 32-bit identifier for the current host as set by `sethostid(2)`.

Files

4008 `/etc/hostid`

sethostname

Name

4009 `sethostname` — set host name

Synopsis

```
4010 #include <unistd.h>
4011 | #include <sys/param.h>
```

```

4012 | #include <sys/utsname.h>
4013 | int sethostname(const char *name, size_t len);

```

Description

4014 | If the process has appropriate privileges, the `sethostname` function shall change the host name of for the
 4015 | current process or machine. The `name` shall point to a null-terminated string of at most `len` bytes that holds the new
 4016 | hostname.

4017 | If the symbol `HOST_NAME_MAX` is defined, or if `sysconf(_SC_HOST_NAME_MAX)` returns a value greater than 0, this
 4018 | value shall represent the maximum length of the new hostname. Otherwise, if the symbol `MAXHOSTLEN` is defined, this
 4019 | value shall represent the maximum length for the new hostname. If none of these values are defined, the maximum
 4020 | length shall be the size of the `nodename` field of the `utsname` structure.

Return Value

4021 | On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

4022 | `EINVAL`

4023 | `len` is negative or larger than the maximum allowed size.

4024 | `EPERM`

4025 | ~~the caller was not the superuser.~~

4026 | the process did not have appropriate privilege.

4027 | `EFAULT`

4028 | `name` is an invalid address.

Notes

4029 | ~~The Single UNIX Specification, Version 2 guarantees that:~~

4030 | ~~Host names are limited to 255 bytes.~~

Rationale

4031 | ISO POSIX (2003) guarantees that:

4032 | Maximum length of a host name (not including the terminating null) as returned from the `gethostname` function shall be at
 4033 | least 255 bytes.

4034 | The glibc C library does not currently define `HOST_NAME_MAX`, and although it provides the name
 4035 | `_SC_HOST_NAME_MAX` a call to `sysconf` returns -1 and does not alter `errno` in this case (indicating that there is no
 4036 | restriction on the hostname length). However, the glibc manual indicates that some implementations may have
 4037 | `MAXHOSTNAMELEN` as a means of detecting the maximum length, while the Linux kernel at release 2.4 and 2.6 stores
 4038 | this hostname in the `utsname` structure. While the glibc manual suggests simply shortening the name until
 4039 | `sethostname` succeeds, the LSB requires that one of the first four mechanisms works. Future versions of glibc may
 4040 | provide a more reasonable result from `sysconf(_SC_HOST_NAME_MAX)`.

setsockopt

Name

4041 `setsockopt` — set options on sockets

Synopsis

```
4042 #include <sys/socket.h>
4043 #include <netinet/in.h>
4044 int setsockopt(int sockfd, int level, int optname, void *optval, socklen_t optlen);
```

Description

4045 In addition to the `setsockopt` options specified in SUSv3, `setsockopt` also supports the options specified here.

4046 The following `setsockopt` operations are provided for `level IPPROTO_IP`:

4047 `IP_MULTICAST_TTL`

4048 Set or reads the time-to-live value of outgoing multicast packets for this socket. *optval* is a pointer to an integer
4049 which contains the new TTL value.

4050 `IP_MULTICAST_LOOP`

4051 Sets a boolean flag indicating whether multicast packets originating locally should be looped back to the local
4052 sockets. *optval* is a pointer to an integer which contains the new flag value.

4053 `IP_ADD_MEMBERSHIP`

4054 Join a multicast group. *optval* is a pointer to a `ip_mreq` structure. Before calling, the caller should fill in the
4055 *imr_multiaddr* field with the multicast group address and the *imr_address* field with the address of the
4056 local interface. If *imr_address* is set to `INADDR_ANY`, then an appropriate interface is chosen by the
4057 system.

4058 `IP_DROP_MEMBERSHIP`

4059 Leave a multicast group. *optval* is a pointer to a `ip_mreq` structure containing the same values as were used
4060 with `IP_ADD_MEMBERSHIP`.

4061 `IP_MULTICAST_IF`

4062 Set the local device for a multicast socket. *optval* is a pointer to a `ip_mreq` structure initialized in the same
4063 manner as with `IP_ADD_MEMBERSHIP`.

4064 The `ip_mreq` structure contains two `struct in_addr` fields: *imr_multiaddr* and *imr_address*.

Return Value

4065 On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.

setutent

Name

4066 | `setutent` — access ~~utmp file~~ user accounting database entries

Synopsis

```
4067 #include <utmp.h>
4068 void setutent(void);
```

Description

4069 | `setutent` rewinds the file pointer to the beginning of the `utmp` file. It is generally a *Good Idea* to call it before any of
 4070 | the other functions.

Errors

4071 | On error, `(struct utmp*)0` will be returned.

Files

4072 | ~~`/var/run/utmp` database of currently logged in users~~ `/var/log/wtmp` database of past user logins

4073 | The `setutent` function shall reset the user accounting database such that the next call to `getutent` shall be return the
 4074 | first record in the database. It is recommended to call it before any of the other functions that operate on the user
 4075 | accounting databases (e.g. `getutent`)

Return Value

4076 | None.

sigandset

Name

4077 sigandset — build a new signal set by combining the two input sets using logical AND

Synopsis

```
4078 #include <signal.h>
4079 | extern int sigandset(sigset_t *set, const sigset_t *left, const sigset_t *right);
```

Description

4080 The `sigandset` is a shall combine the two `signal` function that builds a new signal set sets referenced by combining
 4081 the two input sets `left` and `right`, using a logical AND operation, and shall place the result in the location
 4082 referenced by `set`. The resulting signal set shall contain only signals that are in both the set referenced by `left` and
 4083 the set referenced by `right`.

Return Value

4084 On success, `sigandset` shall return 0. Otherise, `sigandset` shall return -1 and set `errno` to indicate the error.

Errors

4085 EINVAL
 4086 One or more of `set`, `left`, or `right` was a null pointer.

See Also

4087 sigorset

sigblock

Name

4088 sigblock — manipulate the signal mask

Synopsis

```
4089 | #include _BSD_SOURCE
```

```

4090 #include <signal.h>
4091 int sigblock(int mask);

```

Description

```

4092 sigblock is made obsolete by sigprocmask(2).
4093 sigblock adds the signals specified in mask to the set of signals currently being blocked from delivery.

```

Notes

```

4094 Prototype for sigblock is only available if _BSD_SOURCE is defined before the inclusion of any system.
4095 The sigblock function shall add the signals corresponding to the bits set in mask to the set of signals currently being
4096 blocked from delivery.

```

Return Value

```

4097 The sigblock function shall return the previous signal mask.

```

Errors

```

4098 None.

```

Notes

```

4099 sigblock is made obsolete by sigprocmask(2). A future version of this specification may deprecate this function.

```

siggetmask

Name

```

4100 siggetmask — manipulate the signal mask

```

Synopsis

```

4101 #define _BSD_SOURCE
4102 #include <signal.h>
4103 int siggetmask(void);

```

Description

```

4104 The siggetmask function shall return the current set of masked signals.

```

Notes

```

4105 siggetmask is made obsolete by sigprocmask(2).

```


4106 ~~siggetmask returns the current set of masked signals.~~

Notes

4107 ~~Prototype for siggetmask is only available if `_BSD_SOURCE` is defined before the inclusion of any system header~~
 4108 ~~file.~~

sigisemptyset

Name

4109 sigisemptyset — check for empty signal set

Synopsis

4110 #include <signal.h>
 4111 ~~extern int~~ **sigisemptyset**(const sigset_t *set);

Description

4112 The sigisemptyset function shall check for empty signal set referenced by *set*.

Return Value

4113 The sigisemptyset function shall return a positive non-zero value if the signal set referenced by *set* is empty, or
 4114 zero if this set is empty. On error, sigisemptyset shall return -1 and set errno to indicate the error.

Errors

4115 EINVAL
 4116 *set* is a null pointer.

sigorset

Name

4117 sigorset — build a new signal set by combining the two input sets using logical OR

Synopsis

4118 #include <signal.h>
 4119 int **sigorset**(sigset_t *set, const sigset_t *left, const sigset_t *right);

Description

4120 ~~sigisemptyset checks for empty signal set. It returns a non-empty value if *set* is not empty.~~

sigorset

Name

4121 ~~sigorset~~ build a new signal set by combining the two input sets using logical or

Synopsis

4122 ~~#include <signal.h>~~
 4123 ~~extern int sigorset(sigset_t *set, const sigset_t *left, const sigset_t *right);~~

Description

4124 ~~sigorset is a signal function that builds a new signal set by combining the two input sets using logical or.~~

4125 The sigorset shall combine the two signal sets referenced by *left* and *right*, using a logical OR operation, and
 4126 shall place the result in the location referenced by *set*, The resulting signal set shall contain only signals that are in
 4127 either the set referenced by *left* or the set referenced by *right*.

Return Value

4128 On success, sigorset shall return 0. Otherise, sigorset shall return -1 and set errno to indicate the error.

Errors

4129 EINVAL
 4130 One or more of *set*, *left*, or *right* was a null pointer.

See Also

4131 sigorset

sigreturn

Name

4132 sigreturn — return from signal handler and cleanup stack frame

Synopsis

4133 int **sigreturn**(unsigned long __unused);

Description

4134 ~~When the Linux kernel creates the stack frame for a signal handler, a call to sigreturn is inserted into the stack~~
 4135 ~~frame so that the the signal handler will call sigreturn upon return. This inserted call to sigreturn cleans up the~~
 4136 ~~stack so that the process can restart from where it was interrupted by the signal.~~

4137 The sigreturn function is used by the system to cleanup after a signal handler has returned. This function is not in
 4138 the source standard; it is only in the binary standard.

Return Value

4139 sigreturn never returns.

Warning

4140 ~~sigreturn is used by the kernel to implement signal handlers. It should never be called directly. Better yet, the~~
 4141 ~~specific use of __unused varies depending on the architecture.~~

Files

4142 /usr/src/linux/arch/i386/kernel/signal.c

4143 /usr/src/linux/arch/alpha/kernel/entry.s

stime

Name

4144 stime — set time

Synopsis

4145 #define _SVID_SOURCE ~~/* glibc needs this */~~

```

4146 #include <time.h>
4147 int stime(time_t *t);

```

Description

4148 ~~stime sets the system's idea of the time and date. Time, pointed to by *t*, is measured in seconds from 00:00:00 GMT~~
 4149 ~~January 1, 1970. stime may only be executed by the super user.~~

Return Value

4150 ~~On success, 0 is returned. On error, -1 is returned and the global variable `errno` is set appropriately.~~

Errors

4151 ~~EPERM~~

4152 ~~—— The caller is not the super user.~~

Notes

4153 ~~Under *glibc2*, `time.h` only provides a prototype when `_SVID_SOURCE` is defined.~~

4154 If the process has appropriate privilege, the `stime` function shall set the system's idea of the time and date. Time,
 4155 referenced by *t*, is measured in seconds from the epoch (defined in ISO POSIX (2003) as 00:00:00 UTC January 1,
 4156 1970).

Return Value

4157 On success, `stime` shall return 0. Otherwise, `stime` shall return -1 and `errno` shall be set to indicate the error.

Errors

4158 EPERM

4159 The process does not have appropriate privilege.

4160 EINVAL

4161 *t* is a null pointer.

strcpy

Name

4162 `strcpy` — copy a string returning a pointer to its end

Synopsis

```
4163 #include <string.h>
4164 | char *strcpy(char * restrict dest, const char * restrict src);
```

Description

4165 | The `strcpy` ~~copies~~ function shall copy the string pointed to by *src* (including the terminating '\0' character) to the
 4166 array pointed to by *dest*. The strings may not overlap, and the destination string *dest* shall be large enough to
 4167 receive the copy.

Return Value

4168 `strcpy` returns a pointer to the end of the string *dest* (that is, the address of the terminating '\0' character) rather than
 4169 the beginning.

Example

4170 This program uses `strcpy` to concatenate `foo` and `bar` to produce `foobar`, which it then prints.

```
4171 #include <string.h>
4172
4173 int
4174 main (void)
4175 {
4176 |     char buffer[256];
4177     char *to = buffer;
4178     to = strcpy (to, "foo");
4179     to = strcpy (to, "bar");
4180     printf ("%s\n", buffer);
4181 }
```

stpncpy

Name

4182 `stpncpy` — copy a fixed-size string, returning a pointer to its end

Synopsis

```
4183    #include <string.h>
4184    char *stpncpy(char * restrict dest, const char * restrict src, size_t n);
```

Description

4185 The `stpncpy` function shall copy at most *n* characters from the string pointed to by *src*, including the terminating `\0`
 4186 character, to the array pointed to by *dest*. Exactly *n* characters are written at *dest*. If the length `strlen(src)` is
 4187 smaller than *n*, the remaining characters in *dest* are filled with `\0` characters. If the length `strlen(src)` is greater
 4188 than or equal to *n*, *dest* will not be `\0` terminated.

4189 The strings may not overlap.

4190 The programmer shall ensure that there is room for at least *n* characters at *dest*.

Return Value

4191 The `stpncpy` function shall return a pointer to the terminating `NULL` in *dest*, or, if *dest* is not `NULL`-terminated,
 4192 *dest* + *n*.

strcasestr

Name

4193 `strcasestr` — locate a substring ignoring case

Synopsis

```
4194 #include <string.h>
4195 char *strcasestr(const char *s1, const char *s2);
```

Description

4196 ~~strncpy copies at most *n* characters from the string pointed to by *src*, including the terminating `\0` character, to the~~
 4197 ~~array pointed to by *dest*. Exactly *n* characters are written at *dest*. If the length `strlen(src)` is smaller than *n*, the~~
 4198 ~~remaining characters in *dest* are filled with `\0` characters. If the length `strlen(src)` is greater than or equal to *n*,~~
 4199 ~~*dest* will not be `\0` terminated.~~

4200 The strings may not overlap.

4201 The programmer shall ensure that there is room for at least *n* characters at *dest*.

Return Value

4202 ~~strncpy returns a pointer to the terminating `\0` in *dest*, or, if *dest* is not `\0` terminated, *dest* + *n*.~~

strcasestr

Name

4203 ~~strcasestr~~ — locate a substring — ignores the case of both strings

Synopsis

```
4204 #include <string.h>
4205 char *strcasestr(const char *haystack, const char *needle);
```

Description

4206 ~~strcasestr is similar to `strstr`, but ignores the case of both strings.~~

4207 The `strcasestr` shall behave as `strstr`, except that it shall ignore the case of both strings. The `strcasestr`
 4208 function shall be locale aware; that is `strcasestr` shall behave as if both strings had been converted to lower case in
 4209 the current locale before the comparison is performed.

Return Value

4210 Upon successful completion, `strcasestr` shall return a pointer to the located string or a null pointer if the string is
 4211 not found. If *s2* points to a string with zero length, the function shall return *s1*.

strerror_r

Name

4212 `strerror_r` — reentrant version of `strerror`

Synopsis

```
4213 #include <string.h>
4214 | extern char *strerror_r(int errnum, char *buf, size_t buflen);
```

Description

4215 `strerror_r` is a reentrant version of `strerror`. `strerror_r` returns a pointer to an error message corresponding to
 4216 error number *errnum*. The returned pointer may point within the buffer *buf* (at most *buflen* bytes).[†]

Notes

4217 ~~†~~ Note the optional use of the buffer, unlike the `strerror_r` found in the *Single UNIX Specification, Version 3* ISO
 4218 POSIX (2003), in which the message is always copied into the supplied buffer. The return types also differ.

strfry

Name

4220 `strfry` — randomize a string

Synopsis

```
4221 #include <string.h>
4222 char *strfry(char *string);
```

Description

4223 `strfry` randomizes the contents of *string* by using `rand(3)` to randomly swap characters in the string. The result is
 4224 an anagram of *string*.

Return Value

4225 `strfry` returns a pointer to the randomized string.

strndup

Name

4226 `strndup` — return a malloc'd copy of at most the specified number of bytes of a string

Synopsis

```
4227 #include <string.h>
4228 extern char *strndup(const char *string, size_t n);
```

Description

4229 The `strndup` ~~returns~~ function shall return a malloc'd copy of at most *n* bytes of *string*. The resultant string is shall
 4230 be terminated even if no NULL terminator appears before ~~STRING[N]~~`string+n`.

Return Value

4231 On success, `strndup` shall return a pointer to a newly allocated block of memory containing a copy of at most *n* bytes
 4232 of *string*. Otherwise, `strndup` shall return NULL and set `errno` to indicate the error.

Errors

4233 `ENOMEM`
 4234 Insufficient memory available.

strnlen

Name

4235 `strnlen` — determine the length of a fixed-size string

Synopsis

```
4236 #include <string.h>
4237 size_t strnlen(const char *s, size_t maxlen);
```

Description

4238 `strnlen` returns the number of characters in the string *s*, not including the terminating `\0` character, but at most
 4239 *maxlen*. In doing this, `strnlen` looks only at the first *maxlen* characters at *s* and never beyond *s + maxlen*.

Return Value

4240 `strnlen` returns `strlen(s)`, if that is less than *maxlen*, or *maxlen* if there is no `\0` character among the first
 4241 *maxlen* characters pointed to by *s*.

strptime

Name

4242 `strptime` — parse a time string

Description

4243 | The `strptime` ~~is~~shall behave as specified in the *Single UNIX Specification, Version 2* ~~ISO POSIX (2003)~~ with
4244 | differences as listed below.

Number of leading zeroes may be limited

4245 |
4246 | The *Single UNIX Specification, Version 2* ~~ISO POSIX (2003)~~ specifies fields for which "leading zeros are permitted
4247 | but not required"; however, applications shall not expect to be able to supply more leading zeroes for these fields than
4248 | would be implied by the range of the field. Implementations may choose to either match an input with excess leading
4249 | zeroes, or treat this as a non-matching input. For example, `%j` has a range of 001 to 366, so 0, 00, 000, 001, and 045
4250 | are acceptable inputs, but inputs such as 0000, 0366 and the like are not.

Rationale

4251 | *glibc* developers consider it appropriate behavior to forbid excess leading zeroes. When trying to parse a given input
4252 | against several format strings, forbidding excess leading zeroes could be helpful. For example, if one matches
4253 | 0011-12-26 against `%m-%d-%Y` and then against `%Y-%m-%d`, it seems useful for the first match to fail, as it would be
4254 | perverse to parse that date as November 12, year 26. The second pattern parses it as December 26, year 11.
4255 | The *Single UNIX Specification* ~~ISO POSIX (2003)~~ is not explicit that an unlimited number of leading zeroes are
4256 | required, although it may imply this. The LSB explicitly allows implementations to have either behavior. Future
4257 | versions of this standard may require implementations to forbid excess leading zeroes.
4258 | An Interpretation Request is currently pending against ISO POSIX (2003) for this matter.

strsep

Name

4259 strsep — extract token from string

Synopsis

```
4260 #include <string.h>
4261 char *strsep(char **stringp, const char *delim);
```

Description

4262 ~~If~~The strsep function shall find the first token in the string referenced by the pointer *stringp*, using the characters
 4263 in *delim* as delimiters.
 4264 If *stringp* is NULL, strsep ~~returns~~shall return NULL and does nothing else.
 4265 If *stringp* is non-NULL, strsep ~~finds~~shall find the first token in the string referenced by *stringp*, where tokens
 4266 are delimited by ~~symbol~~characters in the string *delim*. This token ~~is~~shall be terminated with a \0 character (by
 4267 overwriting the delimiter), and *stringp* ~~is~~shall be updated to point past the token. In case no delimiter was found,
 4268 the token is taken to be the entire string referenced by *stringp*, and the location referenced by *stringp* is made
 4269 NULL.

Return Value

4270 strsep ~~returns~~shall return a pointer to the token, that is, it returns the original value beginning of *stringp* the token.

Notes

4271 ~~strsep~~The strsep function was introduced as a replacement for strtok, since the latter cannot handle empty fields.
 4272 However, strtok conforms to ~~ANSI~~CISO C (1999) and to ISO POSIX (2003) and hence is more portable.

Bugs

~~strsep suffers from the same problems as~~See Also

4273 strtok. In particular, strsep modifies the original string. Avoid it, strtok_r.

strsignal

Name

4274 strsignal — return string describing signal

Synopsis

```
4275 #define _GNU_SOURCE
```

```

4276 #include <string.h>
4277 char *strsignal(int sig);

4278 extern const char * const sys_siglist[];

```

Description

4279 | The `strsignal` ~~returns~~ function shall return a pointer to a string describing the signal number `sig`. The string can
 4280 | only be used until the next call to `strsignal`.

4281 | The array `sys_siglist` holds the signal description strings indexed by signal number. ~~`strsignal`~~ This array should
 4282 | not be used if possible instead of this array accessed directly by applications.

Return Value

4283 | If `sig` is a valid signal number, `strsignal` ~~returns~~ shall return a pointer to the appropriate description string, or an
 4284 | Otherwise, `strsignal` shall return either a pointer to the string "unknown signal message if the signal number is
 4285 | invalid. On some systems (but not on Linux), a NULL pointer may be-", or a null pointer.

4286 | Although the function is not declared as returning a pointer to a constant character string, applications shall not modify
 4287 | the returned ~~instead for an invalid signal number.~~

strtok_r

Name

4288 ~~strtok_r~~ — extract tokens from strings

Synopsis

4289 ~~#include <string.h>~~
 4290 ~~char ***strtok_r**(char **s*, const char **delim*, char ***ptrptr*);~~

Description

4291 ~~strtok_r~~ parses the string *s* into tokens.¹ The first call to ~~strtok_r~~ should have *s* as its first argument. Subsequent
 4292 calls should have the first argument set to NULL. Each call returns a pointer to the next token, or NULL when no more
 4293 tokens are found.

4294 If a token ends with a delimiter, this delimiting character is overwritten with a \0 and a pointer to the next character is
 4295 saved for the next call to ~~strtok_r~~. The delimiter string *delim* may be different for each call.

4296 ~~ptrptr~~ is a user allocated *char** pointer. It shall be the same while parsing the same string.

Bugs

4297 Never use this function. Note that:

- 4298 • It modifies its first argument.
- 4299 • The identity of the delimiting character is lost.
- 4300 • It cannot be used on constant strings.

Return Value

4301 ~~strtok_r~~ returns a pointer to the next token, or NULL if there are no more tokens.

Notes

4303 1. — A token is a nonempty string of characters not occurring in the string *delim*, followed by \0 or by a character
 4304 occurring in *delim*.

strtoq

Name

4305 strtoq — convert string value to a long or quad_t integer

Synopsis

4306 #include <sys/types.h>
 4307 #include <stdlib.h>

```

4308 #include <limits.h>
4309 quad_t strtouq(const char *nptr, char **endptr, int base);

```

Description

4310 `strtouq` converts the string `nptr` to a quad_t value. The conversion is done according to the given base, which shall be
 4311 between 2 and 36 inclusive, or be the special value 0.

4312 `nptr` may begin with an arbitrary amount of white space (as determined by `isspace(3)`), followed by a single
 4313 optional + or - sign character. If `base` is 0 or 16, the string may then include a 0x prefix, and the number will be read
 4314 in base 16; otherwise, a 0 base is taken as 10 (decimal), unless the next character is 0, in which case it is taken as 8
 4315 (octal).

4316 The remainder of the string is converted to a long value in the obvious manner, stopping at the first character which is
 4317 not a valid digit in the given base. (In bases above 10, the letter A in either upper or lower case represents 10, B
 4318 represents 11, and so forth, with Z representing 35.)

Return Value

4319 `strtouq` returns the result of the conversion, unless the value would underflow or overflow. If an underflow occurs,
 4320 `strtouq` returns `QUAD_MIN`. If an overflow occurs, `strtouq` returns `QUAD_MAX`. In both cases, the global variable
 4321 `errno` is set to `ERANGE`.

Errors

4322 `ERANGE`

4323 The given string was out of range; the value converted has been clamped.

strtouq

Name

4324 `strtouq` — convert a string to an `uquad_t`

Synopsis

```

4325 #include <sys/types.h>
4326 #include <stdlib.h>

```

```

4327 #include <limits.h>
4328 uquad_t strtouq(const char *nptr, char **endptr, int base);

```

Description

4329 `strtouq` converts the string `nptr` to a `uquad_t` value. The conversion is done according to the given base, which shall
 4330 be between 2 and 36 inclusive, or be the special value 0.

4331 `nptr` may begin with an arbitrary amount of white space (as determined by `isspace(3)`), followed by a single
 4332 optional + or - sign character. If `base` is 0 or 16, the string may then include a 0x prefix, and the number will be read
 4333 in base 16; otherwise, a 0 base is taken as 10 (decimal), unless the next character is 0, in which case it is taken as 8
 4334 (octal).

4335 The remainder of the string is converted to an unsigned long value in the obvious manner, stopping at the end of the
 4336 string or at the first character that does not produce a valid digit in the given base. (In bases above 10, the letter A in
 4337 either upper or lower case represents 10, B represents 11, and so forth, with Z representing 35.)

Return Value

4338 On success, `strtouq` returns either the result of the conversion or, if there was a leading minus sign, the negation of
 4339 the result of the conversion, unless the original (non-negated) value would overflow. In the case of an overflow the
 4340 function returns `UQUAD_MAX` and the global variable `errno` is set to `ERANGE`.

Errors

4341 `ERANGE`

4342 The given string was out of range; the value converted has been clamped.

strverscmp

Name

4343 strverscmp — compare strings holding name and indices/version numbers

Synopsis

```
4344 #include <string.h>
4345 extern int strverscmp(const char *s1, const char *s2);
```

Description

4346 ~~strverscmp compares s1 and s2 as~~ The strverscmp function shall compare two strings holding name in a similar
4347 manner to strcmp. If s1 and s2 contain no digits, strverscmp shall behave as strcmp.

4348 The strings are compared by scanning from left to right. If a digit or sequence of digits is encountered in both strings at
4349 the same position, the digit sequence is specially compared, as described below. If the digit sequences compared equal,
4350 the string comparison resumes in both s1 and indices/version numbers s2 after the digit sequence.

4351 Digit sequences are classified as either "integral" or "fractional". A fractional digit sequence begins with a '0';
4352 otherwise the digit sequence shall be treated as an integral digit sequence.

4353 If two integral digit sequences are encountered, they shall be compared as integers for equality. A fractional digit
4354 sequence shall always compare less than an integral digit sequence. If two fractional digit sequences are being
4355 compared, then if the common prefix contains only leading zeroes, the longer part shall compare less than the shorter;
4356 otherwise the comparison shall be strictly numeric.

Examples

4357 Table 1-1. Examples

Call	Return Value
strverscmp("no digit", "no digit")	0 /* same behavior as strcmp */
strverscmp("item#99", "item#100")	<0 /* same prefix, but 99 < 100 */
strverscmp("alpha1", "alpha001")	>0 /* fractional part inferior to integral */
strverscmp("part1_f012", "part1_f01")	>0 /* two fractional parts */
strverscmp("foo.009", "foo.0")	<0 /* two fractional parts but with leading zeroes only */

svc_register

Name

4359 ~~svc_register — Associates program and versnum with the service dispatch procedure, dispatch.~~
 4360 ~~svc_register — Register Remote Procedure Call Interface~~

Synopsis

```
4361 #include <rpc/rpc.h>
4362 void svc_register(SVCXPRT *xprt, u_long prognum, u_long versnum, void (*dispatch)(), u_long
4363 protocol);
```

Description

4364 ~~Associates~~The `svc_register` function shall associate the program identified by *prognum* and at version *versnum*
 4365 with the service dispatch procedure, *dispatch*. If *protocol* is zero, the service is not registered with the portmap
 4366 service. If *protocol* is non-zero, then a mapping of the triple [*prognum*, *versnum*, *protocol*] to
 4367 *xprt->xp_port* is established with the local portmap service (~~generally *protocol* is zero, IPPROTO_UDP or~~
 4368 ~~IPPROTO_TCP~~). The procedure *dispatch* has the following form:

```
4369 int dispatch(request, xprt) struct svc_req * request;-, SVCXPRT * xprt);
```

Return Value

4370 `svc_register` returns 1 if it succeeds, and zero otherwise.

svc_run

Name

4371 `svc_run` — Waits for RPC requests to arrive and calls service procedure.

Synopsis

```
4372 #include <rpc/svc.h>
4373 void svc_run(void);
```

Description

4374 The `svc_run` routine ~~never returns. It waits~~function shall wait for RPC requests to arrive, read and calls-unpack each
 4375 request, and dispatch it to the appropriate service procedure using registered handler. Under normal conditions,
 4376 ~~`svc_getreq` when one arrives. This procedure is usually waiting for a select system call to run~~ shall not return; it
 4377 shall only return if serious errors occur that prevent further processing.

svc_sendreply

Name

4378 `svc_sendreply` — called by RPC service's dispatch routine

Synopsis

4379 `svc_sendreply(SVCXPRT *xprt, xdrproc_t outproc, char out);`

Description

4380 Called by an RPC service's dispatch routine to send the results of a remote procedure call. The parameter *xprt* is the
 4381 request's associated transport handle; *outproc* is the XDR routine which is used to encode the results; and *out* is the
 4382 address of the results. This routine returns one if it succeeds, zero other-wise.

svctcp_create

Name

4383 `svctcp_create` — Creates a TCP/IP-based RPC service transport.

Synopsis

4384 `#include <rpc/rpc.h>`
 4385 `SVCXPRT *svctcp_create(int sock, u_int send_buf_size, u_int recv_buf_size);`

Description

4386 `svctcp_create` creates a TCP/IP-based RPC service transport, to which it returns a pointer. The transport is
 4387 associated with the socket *sock*, which may be `RPC_ANYSOCK`, in which case a new socket is created. If the socket is
 4388 not bound to a local TCP port, then this routine binds it to an arbitrary port. Upon completion, `xprt->xp_sock` is the
 4389 transport's socket descriptor, and `xprt->xp_port` is the transport's port number. Since TCP-based RPC uses buffered
 4390 I/O, users may specify the size of buffers; values of zero choose suitable defaults.

Return Value

4391 `svctcp_create` returns NULL if it fails, or a pointer to the RPC service transport otherwise.

svcudp_create

Name

4392 `svcudp_create` — Creates a UDP-based RPC service transport.

Synopsis

4393 `SVCXPRT *`
4394 `svcudp_create(int sock);`

Description

4395 This call is equivalent to `svcudp_bufcreate (sock, SZ, SZ)` for some default size SZ.

system

Name

4396 `system` — execute a shell command

Synopsis

4397 `#include <stdlib.h>`
 4398 `int system(const char *string);`

Description

4399 The `system` executes a command specified function shall behave as described in *string* by calling `/bin/sh -e`
 4400 *string*, and returns after the command has been completed. During execution of the command, `SIGCHLD` will be
 4401 blocked, and `SIGINT` and `SIGQUIT` will be ignored.

Return Value

4402 The value 127 returned if the `execve` call for `/bin/sh` fails, 1 if there was another error and the return code of the
 4403 command otherwise.
 4404 If the value of *string* is `NULL`, `system` returns a nonzero value if the shell is available, and zero if not.
 4405 `system` does not affect the wait status of any other children ISO POSIX (2003).

Notes

4406 The fact that `system` ignores interrupts is often not what a program wants. The *Single UNIX Specification* ISO POSIX
 4407 (2003) describes some of the consequences; an additional consequence is that a program calling `system` from a loop
 4408 cannot be reliably interrupted. Many programs will want to use the `exec(3)` family of functions instead.
 4409 Do not use `system` from a program with `suid` or `sgid` privileges, because ~~strange~~ unexpected values for some
 4410 environment variables might be used to subvert system integrity. Use the `exec(3)` family of functions instead, but not
 4411 `execlp(3)` or `execvp(3)`. `system` will not, in fact, work properly from programs with `suid` or `sgid` privileges on
 4412 systems on which `/bin/sh` is **bash** version 2, since **bash** 2 drops privileges on startup. (Debian uses a modified **bash**
 4413 which does not do this when invoked as **sh**.)
 4414 The check for the availability of `/bin/sh` is not actually performed; it is always assumed to be available. ~~ISO C~~ ISO C
 4415 (1999) specifies the check, but ~~POSIX.2~~ ISO POSIX (2003) specifies that the return shall always be nonzero, since a
 4416 system without the shell is not conforming, and it is this that is implemented.
 4417 It is possible for the shell command to return 127, so that code is not a sure indication that the `execve` call failed;
 4418 check the global variable `errno` to make sure.

textdomain

Name

4419 | `textdomain` — set the current default message catalog domain

Synopsis

4420 | `#include <libintl.h>`
 4421 | ~~`extern char *textdomain(const char *domainname);`~~

Description

4422 | The `textdomain` ~~sets~~function shall set the current default message catalog domain to `domainname`, ~~which remains~~
 4423 | ~~valid across subsequent~~. Subsequent calls to ~~setlocale~~, and ~~gettext~~.

Return

4424 | ~~On success, textdomain returns the currently selected domain. On error, a NULL pointer is returned~~ and `gettext`
 4425 | ~~use the default message domain.~~
 4426 | If `domainname` is `NULL`, ~~textdomain returns the current default~~the default message domain shall not be altered.
 4427 | If `domainname` is `""`, `textdomain` shall reset the default domain to the system default of "messages".

Return

4428 | On success, `textdomain` shall return the currently selected domain. Otherwise, a null pointer shall be returned, and
 4429 | `errno` set to indicate the error.

Errors

4430 | `ENOMEM`
 4431 | ~~The function may have failed if there was "insufficient"~~Insufficient memory available."

unlink

Name

4432 unlink — remove a directory entry

Synopsis

4433 int **unlink**(const char *path);

Description

4434 unlink is as specified in ~~the ISO/IEC 9945: POSIX (2003-Portable Operating System(POSIX)and The Single~~
 4435 ~~UNIX® Specification(SUS) V3~~), but with differences as listed below.

4436 See also Additional behaviors: unlink/link on directory>.

May return EISDIR on directories

4437 If *path* specifies a directory, the implementation may return EISDIR instead of EPERM as specified by ISO/IEC
 4439 ~~9945: POSIX (2003-Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3)~~.⁺

Notes

1. — Rationale

4442 The Linux kernel has deliberately chosen EISDIR for this case and does not expect to change (Al Viro, personal
 4443 communication).

vasprintf

Name

4444 vasprintf — write formatted output to a string dynamically allocated with malloc and store the address of the
 4445 string

Synopsis

4446 #include <stdarg.h>

```

4447 #include <stdio.h>
4448 | extern int vasprintf(char ** restrict ptr, const char * restrict f_G_format, va_list arg);

```

Description

```

4449 | The vasprintf writes function shall write formatted output to a string dynamically allocated with malloc malloc, and
4450 | stores the address of the that string in the location referenced by ptr. It shall behave as asprintf, except that instead
4451 | of being called with a variable number of arguments, it is called with an argument list as defined by <stdarg.h>.

```

Return Value

```

4452 | Refer to fprintf.

```

Errors

```

4453 | Refer to fprintf.

```

vdprintf

Name

```

4454 | vdprintf — write formatted output to a file descriptor

```

Synopsis

```

4455 | #include <stdio.h>
4456 | extern int vdprintf(int fd, const char * restrict fmt_G_format, va_list arg);

```

Description

```

4457 | The vdprintf writes formatted output to shall behave as vfprintf, except that the first argument is a file descriptor
4458 | rather than a STDIO stream.

```

Return Value

```

4459 | Refer to fprintf.

```

Errors

```

4460 | Refer to fprintf.

```

verrx

Name

4461 | **verrx** — display formatted error messagesmessage and exit

Synopsis

```
4462 | #include <stdarg.h>
4463 | #include <err.h>
4464 | void verrx(int eval, const char *fmt, valist args);
```

Description

4465 | ~~verrx displays a formatted error message on the standard error output. The last component of the program name, a~~
 4466 | ~~colon character, and a space are output. If *fmt* is not NULL, the formatted error message, a colon, and a space are~~
 4467 | ~~output. The output is followed by a newline character.~~
 4468 | void **verrx**(int *eval*, const char **fmt*, va_list *args*);

Description

4469 | The **verrx** shall behave as **errx** except that instead of being called with a variable number of arguments, it is called
 4470 | with an argument list as defined by `<stdarg.h>`.
 4471 | **verrx** does not return, but exits with the value of *eval*.

Return Value

4472 | None.

Errors

4473 | None.

vsyslog

Name

4474 | **vsyslog** — log to system log

Synopsis

```
4475 | #include <stdarg.h>
```



```

4476 | #include <syslog.h>
4477 | void vsyslog(int priority, char *message, va_list arglist);

```

Description

```

4478 | The vsyslog function is identical to syslog as specified in the Single UNIX Specification ISO POSIX (2003), except
4479 | that arglist (as defined by stdarg.h) replaces the variable number of arguments.
4480 | The caller is responsible for running va_end after calling vsyslog.

```

wait3

Name

```

4481 | wait3 — wait for child process

```

Description

```

4482 | wait3 is as specified in the Single UNIX Specification, Version 2, SUSv2 but with differences as listed below.

```

Notes

```

4483 | WCONTINUED and WIFCONTINUED optional

```

```

4484 | Implementations need not support the functionality of WCONTINUED or WIFCONTINUED.

```

wait4

Name

```

4485 | wait4 — wait for process termination, BSD style

```

Synopsis

```

4486 | #include <sys/types.h>
4487 | #include <sys/resource.h>

```

```

4488 #include <sys/wait.h>
4489 pid_t wait4(pid_t pid, int *status, int options, (struct rusage *rusage));

```

Description

4490 `wait4` suspends execution of the current process until a child (as specified by `pid`) has exited, or until a signal is
 4491 delivered whose action is to terminate the current process or to call a signal handling function. If a child (as requested
 4492 by `pid`) has already exited by the time of the call (a so-called "zombie" process), the function returns immediately.
 4493 Any system resources used by the child are freed.

4494 The value of `pid` can be one of:

4495 `< -1`

4496 wait for any child process whose process group ID is equal to the absolute value of `pid`.

4497 `-1`

4498 wait for any child process; this is equivalent to calling `wait3`.

4499 `0`

4500 wait for any child process whose process group ID is equal to that of the calling process.

4501 `> 0`

4502 wait for the child whose process ID is equal to the value of `pid`.

4503 The value of `options` is a bitwise or of zero or more of the following constants:

4504 `WNOHANG`

4505 return immediately if no child is there to be waited for.

4506 `WUNTRACED`

4507 return for children that are stopped, and whose status has not been reported.

4508 If `status` is not `NULL`, `wait4` stores status information in the location `status`. This status can be evaluated with the
 4509 following macros: ⁺

4510 These macros take the `status` value (an `int`) as an argument -- not a pointer to the value!

4511 `WIFEXITED(status)`

4512 is nonzero if the child exited normally.

4513 `WEXITSTATUS(status)`

4514 evaluates to the least significant eight bits of the return code of the child that terminated, which may have been set
 4515 as the argument to a call to `exit` or as the argument for a return statement in the main program. This macro can
 4516 only be evaluated if `WIFEXITED` returned nonzero.

4517 `WIFSIGNALED(status)`

4518 returns true if the child process exited because of a signal that was not caught.

4519 `WTERMSIG(status)`

4520 returns the number of the signal that caused the child process to terminate. This macro can only be evaluated if
 4521 `WIFSIGNALED` returned nonzero.

4522 `WIFSTOPPED(status)`

4523 returns true if the child process that caused the return is currently stopped; this is only possible if the call was
 4524 done using `WUNTRACED`.

4525 `WSTOPSIG(status)`

4526 returns the number of the signal that caused the child to stop. This macro can only be evaluated if `WIFSTOPPED`
 4527 returned nonzero.

4528 If `rusage` is not `NULL`, the struct `rusage` (as defined in `sys/resource.h`) that it points to will be filled with
 4529 accounting information. (See `getrusage(2)` for details.

Return Value

4530 On success, the process ID of the child that exited is returned. On error, -1 is returned (in particular, when no
 4531 unwaited-for child processes of the specified kind exist), or 0 if `WNOHANG` was used and no child was available yet. In
 4532 the latter two cases, the global variable `errno` is set appropriately.

Errors

4533 `ECHILD`

4534 No unwaited-for child process as specified does exist.

4535 `ERESTARTSYS`

4536 A `WNOHANG` was not set and an unblocked signal or a `SIGCHLD` was caught. This error is returned by the system
 4537 call. The library interface is not allowed to return `ERESTARTSYS`, but will return `EINTR`.

Notes

4538 ~~1. These macros take the `stat` buffer (an `int`) as an argument — not a pointer to the buffer!~~

waitpid

Name

4540 `waitpid` — wait for child process

Description

4541 `waitpid` is as specified in ~~the *Single UNIX Specification*~~ ISO POSIX (2003), but with differences as listed below.

Need not support `WCONTINUED` or `WIFCONTINUED`

4543 Implementations need not support the functionality of `WCONTINUED` or `WIFCONTINUED`.

warn

Name

4544 warn — formatted error messages

Synopsis

```
4545 #include <err.h>
4546 void warn(const char *fmt ...);
```

Description

4547 The `warn` displays function shall display a formatted error message on the standard error stream. The output—The shall
 4548 consist of the last component of the program name, a colon character, and a space are output character. If `fmt` is not
 4549 non-NULL, it shall be used as a format string for the `printf` family of functions, and the formatted error message, a
 4550 colon character, and a space are output. The written to `stderr`. Finally, the error message string affiliated with the
 4551 current value of the global variable `errno` is output. The output is shall be written to `stderr`, followed by a newline
 4552 character.

Return Value

4553 None.

Errors

4554 None.

warnx

Name

4555 warnx — formatted error messages

Synopsis

4556 #include <err.h>
 4557 void **warnx**(const char *fmt ...);

Description

4558 The **warnx** displaysfunction shall display a formatted error message on the standard error outputstream. The last
 4559 component of the program name, a colon character, and a space are shall be output. If *fmt* is not non-NULL, it shall be
 4560 used as the format string for the printf family of functions, and the formatted error message, a colon character, and a
 4561 space are shall be output. The output. The output is shall be followed by a newline character.

Return Value

4562 None.

Errors

4563 None.

wcpcpy

Name

4564 `wcpcpy` — copy a wide character string, returning a pointer to its end

Synopsis

```
4565 #include <wchar.h>
4566 wchar_t *wcpcpy(wchar_t *dest, const wchar_t *src);
```

Description

4567 `wcpcpy` is the wide-character equivalent of `strcpy`. It copies the wide character string `src`, including the terminating
4568 `L'\0'` character, to the array `dest`.

4569 The strings may not overlap.

4570 The programmer shall ensure that there is room for at least `wcslen(src)+1` wide characters at `dest`.

Return Value

4571 `wcpcpy` returns a pointer to the end of the wide-character string `dest`, that is, a pointer to the terminating `L'\0'`
4572 character.

wcpncpy

Name

4573 `wcpncpy` — copy a fixed-size string of wide characters, returning a pointer to its end

Synopsis

```
4574 #include <wchar.h>
4575 wchar_t *wcpncpy(wchar_t *dest, const wchar_t *src, size_t n);
```

Description

4576 `wcpncpy` is the wide-character equivalent of `stpncpy`. It copies at most `n` wide characters from the wide-character
4577 string `src`, including the terminating `L'\0'` character, to the array `dest`. Exactly `n` wide characters are written at `dest`.
4578 If the length `wcslen(src)` is smaller than `n`, the remaining wide characters in the array `dest` are filled with `L'\0'`
4579 characters. If the length `wcslen(src)` is greater than or equal to `n`, the string `dest` will not be `L'\0'` terminated.

4580 The strings may not overlap.

4581 The programmer shall ensure that there is room for at least `n` wide characters at `dest`.

Return Value

4582 `wcpncpy` returns a pointer to the wide character one past the last non-null wide character written.

wscasecmp

Name

4583 `wscasecmp` — compare two wide-character strings, ignoring case

Synopsis

```
4584 #include <wchar.h>
4585 int wscasecmp(const wchar_t *s1, const wchar_t *s2);
```

Description

4586 `wscasecmp` is the wide-character equivalent of `strcasecmp`. It compares the wide-character string `s1` and the
 4587 wide-character string `s2`, ignoring case differences (`towupper`, `towlower`).

Return Value

4588 `wscasecmp` returns 0 if the wide-character strings `s1` and `s2` are equal except for case distinctions. It returns a
 4589 positive integer if `s1` is greater than `s2`, ignoring case. It returns a negative integer if `s1` is smaller than `s2`, ignoring
 4590 case.

Notes

4591 The behavior of `wscasecmp` depends upon the `LC_CTYPE` category of the current locale.

wcsdup

Name

4592 `wcsdup` — duplicate a wide-character string

Synopsis

```
4593 #include <wchar.h>
4594 wchar_t *wcsdup(const wchar_t *s);
```

Description

4595 `wcsdup` is the wide-character equivalent of `strdup`. It allocates and returns a new wide-character string whose initial
 4596 contents is a duplicate of the wide-character string `s`.

4597 Memory for the new wide-character string is obtained with `malloc(3)`, and can be freed with `free(3)`.

Return Value

4598 `wcsdup` returns a pointer to the new wide-character string, or `NULL` if sufficient memory was not available.

wcsncasecmp

Name

4599 `wcsncasecmp` — compare two fixed-size wide-character strings, ignoring case

Synopsis

```
4600 #include <wchar.h>
4601
4602 int wcsncasecmp(const wchar_t *s1, const wchar_t *s2, size_t n);
```

Description

4603 `wcsncasecmp` is the wide-character equivalent of `strncasecmp`. It compares the wide-character string *s1* and the
4604 wide-character string *s2*, but at most *n* wide characters from each string, ignoring case differences (`towupper`,
4605 `towlower`).

Return Value

4606 `wcsncasecmp` returns 0 if the wide-character strings *s1* and *s2*, truncated to at most length *n*, are equal except for case
4607 distinctions. It returns a positive integer if truncated *s1* is greater than truncated *s2*, ignoring case. It returns a
4608 negative integer if truncated *s1* is smaller than truncated *s2*, ignoring case.

Notes

4609 The behavior of `wcsncasecmp` depends upon the `LC_CTYPE` category of the current locale.

wcsnlen

Name

4610 `wcsnlen` — determine the length of a fixed-size wide-character string

Synopsis

```
4611 #include <wchar.h>  
4612 size_t wcsnlen(const wchar_t *s, size_t maxlen);
```

Description

4613 `wcsnlen` is the wide-character equivalent of `strlen`. It returns the number of wide-characters in the string *s*, not
4614 including the terminating `L'\0'` character, but at most *maxlen*. In doing this, `wcsnlen` looks only at the first *maxlen*
4615 wide-characters at *s* and never beyond *s + maxlen*.

Return Value

4616 `wcsnlen` returns `wcslen(s)` if that is less than *maxlen*, or *maxlen* if there is no `L'\0'` character among the first
4617 *maxlen* wide characters pointed to by *s*.

Notes

4618 The behavior of `wcsncasecmp` depends on the `LC_CTYPE` category of the current locale.

wcsnrtombs

Name

4619 `wcsnrtombs` — convert a wide character string to a multi-byte string

Synopsis

```
4620 #include <wchar.h>
4621 size_t wcsnrtombs(char *dest, const wchar_t **src, size_t nwc, size_t len, mbstate_t *ps);
```

Description

4622 `wcsnrtombs` is like `wcsrtombs`, except that the number of wide characters to be converted, starting at `src`, is limited
4623 to `nwc`.

4624 If `dest` is not a NULL pointer, `wcsnrtombs` converts at most `nwc` wide characters from the wide-character string
4625 `src` to a multibyte string starting at `dest`. At most `len` bytes are written to `dest`. The state `ps` is updated.

4626 The conversion is effectively performed by repeatedly calling:

```
4627 wctomb(dest, *src, ps)
```

4628 as long as this call succeeds, and then incrementing `dest` by the number of bytes written and `src` by 1.

4629 The conversion can stop for three reasons:

- 4630 • A wide character has been encountered that cannot be represented as a multibyte sequence (according to the current
4631 locale). In this case `src` is left pointing to the invalid wide character, `(size_t)(-1)` is returned, and `errno` is set to
4632 `EILSEQ`.
- 4633 • `nwc` wide characters have been converted without encountering a `L'\0'`, or the length limit forces a stop. In this case,
4634 `src` is left pointing to the next wide character to be converted, and the number bytes written to `dest` is returned.
- 4635 • The wide-character string has been completely converted, including the terminating `L'\0'` (which has the side effect
4636 of bringing back `ps` to the initial state). In this case, `src` is set to `NULL`, and the number of bytes written to `dest`,
4637 excluding the terminating `L'\0'` byte, is returned.

4638 If `dest` is `NULL`, `len` is ignored, and the conversion proceeds as above, except that the converted bytes are not written
4639 out to memory, and that no destination length limit exists.

4640 In both of the above cases, if `ps` is a `NULL` pointer, a static anonymous state only known to `wcsnrtombs` is used
4641 instead.

4642 The programmer shall ensure that there is room for at least `len` bytes at `dest`.

Return Value

4643 `wcsnrtombs` returns the number of bytes that make up the converted part of multibyte sequence, not including the
4644 terminating `L'\0'` byte. If a wide character was encountered which could not be converted, `(size_t)(-1)` is returned, and
4645 the global variable `errno` set to `EILSEQ`.

Notes

4646 The behavior of `wcsnrtombs` depends on the `LC_CTYPE` category of the current locale.
 4647 Passing `NULL` as *ps* is not multi-thread safe.

wcstoq

Name

4648 `wcstoq` — convert ~~initial portion of~~ wide string ~~NPTR~~ to long long int representation

Synopsis

```
4649 #include <wchar.h>
4650 | extern long long int wcstoq(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int
4651 | base);
```

Description

4652 The `wcstoq` ~~converts~~ function shall convert the initial portion of the wide string *nptr* to long long int
 4653 representation. It is identical to `wcstoll`.

Return Value

4654 Refer to `wcstoll`.

Errors

4655 Refer to `wcstoll`.

wstouq

Name

4656 | `wstouq` — convert initial portion of wide string `NPTR` to unsigned long long int representation

Synopsis

```
4657 | #include <wchar.h>
4658 | extern unsigned long long int wstouq(const wchar_t * restrict nptr, wchar_t ** restrict
4659 | endptr, int base);
```

Description

4660 | The `wstouq` converts function shall convert the initial portion of the wide string *nptr* to unsigned long long int
4661 | representation. It is identical to `wstoull`.

Return Value

4662 | Refer to `wstoull`.

Errors

4663 | Refer to `wstoull`.

xdr_u_int

Name

4664 | `xdr_u_int` — library routines for external data representation

Synopsis

```
4665 | int xdr_u_int(XDR * xdrs, unsigned int * up);
```

Description

4666 | `xdr_u_int` is a filter primitive that translates between C unsigned integers and their external representations.

Return Value

4667 | On success, 1 is returned. On error, 0 is returned.

1.5. Interfaces for libm

4668 | Table 1-2829 defines the library name and shared object name for the libm library

Table 1-2829. libm Definition

Library:	libm
SONAME:	See archLSB.

The behavior of the interfaces in this library is specified by the following specifications:

ISO/IEC 9899: C (1999, Programming Language—C)
CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0, €606) SUSv2
ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.5.1. Math

1.5.1.1. Interfaces for Math

An LSB conforming implementation shall provide the generic functions for Math specified in Table 1-2930, with the full functionality as described in the referenced underlying specification.

Table 1-2930. libm - Math Function Interfaces

<code>aacos</code> [1]	<code>expcexp</code> [1]	<code>expfexpf</code> [1]	<code>jnfjnf</code> [2]	<code>fremquof</code> [1]
<code>aacosf</code> [1]	<code>expfcexpf</code> [1]	<code>explexpl</code> [1]	<code>jnljnl</code> [2]	<code>fremquol</code> [1]
<code>acosh</code> [1]	<code>explcexpl</code> [1]	<code>expmlexpm1</code> [1]	<code>ldexpldexp</code> [1]	<code>rintrint</code> [1]
<code>acoshf</code> [1]	<code>imagcimag</code> [1]	<code>fabsfabs</code> [1]	<code>ldexpfldexpf</code> [1]	<code>rintfrintf</code> [1]
<code>acoshl</code> [1]	<code>imagfcimagf</code> [1]	<code>fabssfabsf</code> [1]	<code>ldexplhdexpl</code> [1]	<code>rintlrintl</code> [1]
<code>aacosl</code> [1]	<code>imaglcimagl</code> [1]	<code>fabslfabsl</code> [1]	<code>lgammalgamma</code> [1]	<code>roundround</code> [1]
<code>asin</code> [1]	<code>elogclog</code> [1]	<code>fdimfdim</code> [1]	<code>lgamma_rlgamma_r</code> [2]	<code>roundf</code> [1]
<code>asinf</code> [1]	<code>elog10clog10</code> [2]	<code>fdimffdimf</code> [1]	<code>lgammaflgammaf</code> [1]	<code>roundl</code> [1]
<code>asinh</code> [1]	<code>elog10fclog10f</code> [2]	<code>fdimlfdiml</code> [1]	<code>lgamma_flgammaf_r</code> [2]	<code>scalbscalb</code> [1]
<code>asinhf</code> [1]	<code>elog10lclog10l</code> [2]	<code>fe_clear_exceptfeclear except</code> [1]	<code>lgammallgamma_l</code> [1]	<code>scalbfscalbf</code> [2]
<code>asinhhl</code> [1]	<code>elogfclogf</code> [1]	<code>fe_getenvfegetenv</code> [1]	<code>lgamma_rlgamma_l_r</code> [2]	<code>scalblscalbl</code> [2]
<code>asinhl</code> [1]	<code>eloglclogl</code> [1]	<code>fe_get_exceptionflagfeget exceptionflag</code> [1]	<code>llrintllrint</code> [1]	<code>scalblnscalbln</code> [1]
<code>atan</code> [1]	<code>conjconj</code> [1]	<code>fe_get_roundingfegetroun d</code> [1]	<code>llrintfllrintf</code> [1]	<code>scalblnfscalblnf</code> [1]

atan2atan2 [1]	econjfconjf [1]	feholdexceptfeholdexcept [1]	llrintllrintl [1]	scalbnscalbnl [1]
atan2fatan2f [1]	econjlconjl [1]	feraiseexceptferaiseexcept [1]	llroundllround [1]	scalbnscalbn [1]
atan2latan2l [1]	eopysigncopysign [1]	fesetenvfesetenv [1]	llroundfllroundf [1]	scalbnfscalbnf [1]
atanfatanf [1]	eopysignfcopysignf [1]	fesetexceptflagfesetexceptflag [1]	llroundllroundl [1]	scalbnlscalbnl [1]
atanhatanh [1]	eopysignlcopysignl [1]	fesetroundfesetround [1]	loglog [1]	significandsignificand [2]
atanhfatanhf [1]	eoscosc [1]	fetestexceptfetestexcept [1]	log10log10 [1]	significandsignificandf [2]
atanhlatanhl [1]	eosfcosf [1]	feupdateenvfeupdateenv [1]	log10flog10f [1]	significandsignificandl [2]
atanlatanl [1]	eoshcosh [1]	finitefinite [3]	log10llog10l [1]	sinsin [1]
eabscabs [1]	eoshfcoshf [1]	finiteffinitelf [2]	log1plog1p [1]	sineossincos [2]
eabscabsf [1]	eoshlcoshl [1]	finitel finitel [2]	logblogb [1]	sineosfsincosf [2]
eabscabsl [1]	eoscosl [1]	floorfloor [1]	logflogf [1]	sineoslsincosl [2]
eaeoscacos [1]	epowcpow [1]	floorffloorf [1]	loglogl [1]	sinf sinf [1]
eaeoscacosf [1]	epowfcpowf [1]	floorffloorl [1]	llrintllrint [1]	sinhsinh [1]
eaeoshcacosh [1]	epowlcpowl [1]	fmafma [1]	llrintfllrintf [1]	sinhfsinhf [1]
eaeoshfcacoshf [1]	eprojcproj [1]	fmaffmaf [1]	llrintllrintl [1]	sinhlsinh [1]
eaeoshlcacoshl [1]	eprojfcprojf [1]	fmal fmal [1]	llroundllround [1]	sinhlsinl [1]
eaeoslcacosl [1]	eprojlcproj [1]	fmaxfmax [1]	llroundfllroundf [1]	sqrtsqrt [1]
eargcarg [1]	erealcreal [1]	fmaxffmaxf [1]	llroundllroundl [1]	sqrtsqrtrf [1]
eargfcargf [1]	erealfcrealf [1]	fmaxlfmaxl [1]	matherrmatherr [2]	sqrtsqrtrf [1]
eargfcargl [1]	ereallcreall [1]	fminfmin [1]	modfmodf [1]	tan tan [1]
easincasin [1]	esincsin [1]	fminffminf [1]	modffmodff [1]	tanftanf [1]
easincasinf [1]	esinfcsinf [1]	fminl fminl [1]	modflmodfl [1]	tanh tanh [1]
easinhcasinh [1]	esinhcsinh [1]	fmodfmod [1]	nan nan [1]	tanhftanhf [1]
easinhfcasinhf [1]	esinhfcsinhf [1]	fmodffmodf [1]	nanfnanf [1]	tanhltanh [1]
easinhlcasinh [1]	esinhlcasinh [1]	fmodffmodl [1]	nanlnanl [1]	tanhtanl [1]
easinhlcasinl [1]	esinhlcasinl [1]	frexpfrexp [1]	nearbyintnearbyint	tgammatgamma [1]

			[1]	
catan catan [1]	esqrt csqrt [1]	frexp frexp [1]	nearbyint nearbyintf [1]	tgamma ftgamma [1]
catanf catanf [1]	esqrtf csqrtf [1]	frexpl frexpl [1]	nearbyint nearbyintl [1]	tgamma ftgamma [1]
catanh catanh [1]	esqrtl csqrtl [1]	gamma gamma [3]	nextafter nextafter [1]	trunc trunc [1]
catanhf catanhf [1]	etan ctan [1]	gammaf gammaf [2]	nextafterf nextafterf [1]	truncf truncf [1]
catanhl catanhl [1]	etanf ctanf [1]	gamma gamma [2]	nextafterl nextafterl [1]	trunc trunc [1]
catanl catanl [1]	etanh ctanh [1]	hypot hypot [1]	nexttoward nexttoward [1]	y0 y0 [1]
cbt cbt [1]	etanhf ctanhf [1]	hypotf hypotf [1]	nexttowardf nexttowardf [1]	y0f y0f [2]
cbtf cbtf [1]	etanh lctanh [1]	hypot hypotl [1]	nexttowardl nexttowardl [1]	y0l y0l [2]
cbtrl cbtrl [1]	etan lctanl [1]	ilogb ilogb [1]	pow pow [1]	y1 y1 [1]
ccos ccos [1]	dremf dremf [2]	ilogbf ilogbf [1]	pow10 pow10 [2]	y1f y1f [2]
ccosf ccosf [1]	drem ldrem [2]	ilogbl ilogbl [1]	pow10f pow10f [2]	y1l y1l [2]
ccosh ccosh [1]	erf erf [1]	j0 j0 [1]	pow10l pow10l [2]	yn yn [1]
ccoshf ccoshf [1]	erfe erfc [1]	j0f j0f [2]	powf powf [1]	ynf ynf [2]
ccoshl ccoshl [1]	erfe erfcf [1]	j0l j0l [2]	powl powl [1]	ynl ynl [2]
ccosl ccosl [1]	erfe erfc [1]	j1 j1 [1]	remainder remainder [1]	
ceil ceil [1]	erff erff [1]	j1f j1f [2]	remainderf remainderf [1]	
ceilf ceilf [1]	erfl erfl [1]	j1l j1l [2]	remainderl remainderl [1]	
ceil ceil [1]	exp exp [1]	j1n j1n [1]	remquo remquo [1]	

Referenced Specification(s)

- [1]. ISO/IEC 9945: POSIX (2003-Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)
- [2]. ISO/IEC 9899: C (1999, Programming Language—C)

[3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, €606) SUSv2

An LSB conforming implementation shall provide the generic data interfaces for Math specified in Table 1-3031, with the full functionality as described in the referenced underlying specification.

Table 1-3031. libm - Math Data Interfaces

signgam	signgam [1]			
---------	-------------	--	--	--

Referenced Specification(s)

[1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.6. Data Definitions for libm

This section defines global identifiers and their values that are associated with interfaces contained in libm. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content.

These definitions are intended to supplement those provided in the referenced underlying specifications.

This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are specified in ISO 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.

1.6.1. complex.h

```
#define complex _Complex
```

1.6.2. math.h

```
#define DOMAIN 1
#define SING 2

struct exception
{
    int type;
    char *name;
    double arg1;
    double arg2;
    double retval;
};

#define isinf(x) ((sizeof (x) == sizeof (float) ? __isinf (x) : sizeof (x) == sizeof (double) ? __isinf (x) : __isinfl (x)))
#define isnan(x) ((sizeof (x) == sizeof (float) ? __isnan (x) : sizeof (x) == sizeof (double) ? __isnan (x) : __isnani (x)))
```



```
4718 #define HUGE_VAL          0x1.0p2047
4719 #define HUGE_VALF         0x1.0p255f
4720 #define HUGE_VALL         0x1.0p32767L
4721
4722 #define NAN                ((float)0x7fc00000UL)
4723 #define M_1_PI             0.31830988618379067154
4724 #define M_LOG10E           0.43429448190325182765
4725 #define M_2_PI             0.63661977236758134308
4726 #define M_LN2              0.69314718055994530942
4727 #define M_SQRT1_2          0.70710678118654752440
4728 #define M_PI_4             0.78539816339744830962
4729 #define M_2_SQRTPI         1.12837916709551257390
4730 #define M_SQRT2            1.41421356237309504880
4731 #define M_LOG2E            1.4426950408889634074
4732 #define M_PI_2             1.57079632679489661923
4733 #define M_LN10             2.30258509299404568402
4734 #define M_E                2.7182818284590452354
4735 #define M_PI               3.14159265358979323846
4736 #define INFINITY           HUGE_VALF
4737
4738 #define MATH_ERRNO         1
4739 #define MATH_ERREXCEPT  2
```

1.7. Interfaces for libpthread

4740 Table 1-3432 defines the library name and shared object name for the libpthread library

4741 **Table 1-3432. libpthread Definition**

Library:	libpthread
SONAME:	libpthread.so.0

4743 The behavior of the interfaces in this library is specified by the following specifications:

- Large File Support
- ~~Linux Standard Base~~this specification
- ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

1.7.1. Realtime Threads

1.7.1.1. Interfaces for Realtime Threads

4746 No external functions are defined for libpthread - Realtime Threads

1.7.2. Advanced Realtime Threads

1.7.2.1. Interfaces for Advanced Realtime Threads

4748 No external functions are defined for libpthread - Advanced Realtime Threads

1.7.3. Posix Threads

1.7.3.1. Interfaces for Posix Threads

An LSB conforming implementation shall provide the generic functions for Posix Threads specified in Table 1-3233, with the full functionality as described in the referenced underlying specification.

Table 1-3233. libpthread - Posix Threads Function Interfaces

pthread_cleanup_pop pthread_cleanup_pop [1]	pthread_cancel pthread_cancel [2]	pthread_join pthread_join [2]	pthread_rwlock_destroy pthread_rwlock_destroy [2]	pthread_setconcurrency pthread_setconcurrency [2]
pthread_cleanup_push pthread_cleanup_push [1]	pthread_cond_broadcast pthread_cond_broadcast [2]	pthread_key_create pthread_key_create [2]	pthread_rwlock_init pthread_rwlock_init [2]	pthread_setspecific pthread_setspecific [2]
pthread_read pthread_read [2]	pthread_cond_destroy pthread_cond_destroy [2]	pthread_key_delete pthread_key_delete [2]	pthread_rwlock_rdlock pthread_rwlock_rdlock [2]	pthread_sigmask pthread_sigmask [2]
pthread_read64 pthread_read64 [3]	pthread_cond_init pthread_cond_init [2]	pthread_kill pthread_kill [2]	pthread_rwlock_timedrdlock pthread_rwlock_timedrdlock [2]	pthread_testcancel pthread_testcancel [2]
pthread_attr_destroy pthread_attr_destroy [2]	pthread_cond_signal pthread_cond_signal [2]	pthread_mutex_destroy pthread_mutex_destroy [2]	pthread_rwlock_timedwrlock pthread_rwlock_timedwrlock [2]	pthread_write pthread_write [2]
pthread_attr_getdetachstate pthread_attr_getdetachstate [2]	pthread_cond_timedwait pthread_cond_timedwait [2]	pthread_mutex_init pthread_mutex_init [2]	pthread_rwlock_tryrdlock pthread_rwlock_tryrdlock [2]	pthread_write64 pthread_write64 [3]
pthread_attr_getguardsize pthread_attr_getguardsize [2]	pthread_cond_wait pthread_cond_wait [2]	pthread_mutex_lock pthread_mutex_lock [2]	pthread_rwlock_trywrlock pthread_rwlock_trywrlock [2]	sem_close sem_close [2]
pthread_attr_getschedparam pthread_attr_getschedparam [2]	pthread_condattr_destroy pthread_condattr_destroy [2]	pthread_mutex_trylock pthread_mutex_trylock [2]	pthread_rwlock_unlock pthread_rwlock_unlock [2]	sem_destroy sem_destroy [2]
pthread_attr_getstackaddr pthread_attr_getstackaddr [2]	pthread_condattr_getshared pthread_condattr_getshared [2]	pthread_mutex_unlock pthread_mutex_unlock [2]	pthread_rwlock_wrlock pthread_rwlock_wrlock [2]	sem_getvalue sem_getvalue [2]
pthread_attr_getstacksize pthread_attr_getstacksize [2]	pthread_condattr_init pthread_condattr_init [2]	pthread_mutexattr_destroy pthread_mutexattr_destroy [2]	pthread_rwlockattr_destroy pthread_rwlockattr_destroy [2]	sem_init sem_init [2]
pthread_attr_init pthread_attr_init [2]	pthread_condattr_set pthread_condattr_set [2]	pthread_mutexattr_set pthread_mutexattr_set [2]	pthread_rwlockattr_set pthread_rwlockattr_set [2]	sem_open sem_open [2]

read_attr_init [2]	pthread_condattr_setpshared [2]	getpsharedpthread_mutexattr_getpshared [2]	getpsharedpthread_rwlockattr_getpshared [2]	[2]
pthread_attr_setdetachstate pthread_attr_setdetachstate [2]	pthread_createpthread_create [2]	pthread_mutexattr_gettypepthread_mutexattr_gettype [2]	pthread_rwlockattr_initpthread_rwlockattr_init [2]	sem_postsem_post [2]
pthread_attr_setguardsize pthread_attr_setguardsize [2]	pthread_detachpthread_detach [2]	pthread_mutexattr_initpthread_mutexattr_init [2]	pthread_rwlockattr_setpsharedpthread_rwlockattr_setpshared [2]	sem_timedwaitsem_timedwait [2]
pthread_attr_setschedparam pthread_attr_setschedparam [2]	pthread_equalpthread_equal [2]	pthread_mutexattr_setpsharedpthread_mutexattr_setpshared [2]	pthread_selfpthread_self [2]	sem_trywaitsem_trywait [2]
pthread_attr_setstackaddr pthread_attr_setstackaddr [2]	pthread_exitpthread_exit [2]	pthread_mutexattr_settypepthread_mutexattr_settype [2]	pthread_setcancelstatepthread_setcancelstate [2]	sem_unlinksem_unlink [2]
pthread_attr_setstacksize pthread_attr_setstacksize [2]	pthread_getspecificpthread_getspecific [2]	pthread_oncepthread_once [2]	pthread_setcanceltypepthread_setcanceltype [2]	sem_waitsem_wait [2]

4753

4754 *Referenced Specification(s)*4755 [1]. ~~Linux Standard Base~~this specification4756 [2]. ISO/IEC 9945: POSIX (2003 ~~Portable Operating System (POSIX)~~and The Single UNIX® Specification(SUS)4757 ~~V3~~)

4758 [3]. Large File Support

1.8. Data Definitions for libpthread

4759 This section defines global identifiers and their values that are associated with interfaces contained in libpthread.

4760 These definitions are organized into groups that correspond to system headers. This convention is used as a
4761 convenience for the reader, and does not imply the existence of these headers, or their content.

4762 These definitions are intended to supplement those provided in the referenced underlying specifications.

4763 This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are
4764 specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of
4765 these data objects does not preclude their use by other programming languages.

1.8.1. pthread.h

4766

4767 #define PTHREAD_MUTEX_DEFAULT 1

4768 #define PTHREAD_MUTEX_NORMAL 1

4769 #define PTHREAD_MUTEX_TIMED_NP 1

```

4770 #define PTHREAD_MUTEX_RECURSIVE 2
4771 #define PTHREAD_RWLOCK_DEFAULT_NP 2
4772 #define PTHREAD_MUTEX_ERRORCHECK 3
4773 #define pthread_cleanup_pop(execute) _pthread_cleanup_pop(& _buffer, (execute));}
4774 #define __LOCK_INITIALIZER { 0, 0 }
4775 #define PTHREAD_RWLOCK_INITIALIZER { __LOCK_INITIALIZER, 0, NULL, NULL,
4776 NULL, PTHREAD_RWLOCK_DEFAULT_NP, PTHREAD_PROCESS_PRIVATE }
4777 #define PTHREAD_MUTEX_INITIALIZER { 0, 0, 0, PTHREAD_MUTEX_TIMED_NP, __LOCK_INITIALIZER }
4778 #define pthread_cleanup_push(routine, arg) {struct _pthread_cleanup_buffer
4779 _buffer; pthread_cleanup_push(& _buffer, (routine), (arg));}
4780 #define PTHREAD_COND_INITIALIZER {__LOCK_INITIALIZER, 0}
4781
4782 struct _pthread_cleanup_buffer
4783 {
4784     void (*__routine) (void *);
4785     void *__arg;
4786     int __canceltype;
4787     struct _pthread_cleanup_buffer *__prev;
4788 }
4789 ;
4790 typedef unsigned int pthread_key_t;
4791 typedef int pthread_once_t;
4792 typedef long long __pthread_cond_align_t;
4793
4794 typedef unsigned long pthread_t;
4795 struct _pthread_fastlock
4796 {
4797     long __status;
4798     int __spinlock;
4799 }
4800 ;
4801
4802 typedef struct _pthread_descr_struct *pthread_descr;
4803
4804 typedef struct
4805 {
4806     int __m_reserved;
4807     int __m_count;
4808     pthread_descr __m_owner;
4809     int __m_kind;
4810     struct _pthread_fastlock __m_lock;
4811 }
4812 pthread_mutex_t;
4813 typedef struct
4814 {
4815     int __mutexkind;
4816 }
4817 pthread_mutexattr_t;
4818
4819 typedef struct
4820 {
4821     int __detachstate;
4822     int __schedpolicy;

```

```

4823     struct sched_param __schedparam;
4824     int __inheritsched;
4825     int __scope;
4826     size_t __guardsize;
4827     int __stackaddr_set;
4828     void *__stackaddr;
4829     unsigned long __stacksize;
4830 }
4831 pthread_attr_t;
4832
4833 typedef struct
4834 {
4835     struct _pthread_fastlock __c_lock;
4836     _pthread_descr __c_waiting;
4837     char __padding[48 - sizeof (struct _pthread_fastlock) -
4838             sizeof (_pthread_descr) - sizeof (__pthread_cond_align_t)];
4839     _pthread_cond_align_t __align;
4840 }
4841 pthread_cond_t;
4842 typedef struct
4843 {
4844     int __dummy;
4845 }
4846 pthread_condattr_t;
4847
4848 typedef struct _pthread_rwlock_t
4849 {
4850     struct _pthread_fastlock __rw_lock;
4851     int __rw_readers;
4852     _pthread_descr __rw_writer;
4853     _pthread_descr __rw_read_waiting;
4854     _pthread_descr __rw_write_waiting;
4855     int __rw_kind;
4856     int __rw_pshared;
4857 }
4858 pthread_rwlock_t;
4859 typedef struct
4860 {
4861     int __lockkind;
4862     int __pshared;
4863 }
4864 pthread_rwlockattr_t;
4865
4866 #define PTHREAD_CREATE_JOINABLE 0
4867 #define PTHREAD_INHERIT_SCHED 0
4868 #define PTHREAD_ONCE_INIT 0
4869 #define PTHREAD_PROCESS_PRIVATE 0
4870 #define PTHREAD_CREATE_DETACHED 1
4871 #define PTHREAD_EXPLICIT_SCHED 1
4872 #define PTHREAD_PROCESS_SHARED 1
4873
4874 #define PTHREAD_CANCELED ((void*)-1)
4875 #define PTHREAD_CANCEL_DEFERRED 0

```

```

4876 #define PTHREAD_CANCEL_ENABLE    0
4877 #define PTHREAD_CANCEL_ASYNCHRONOUS    1
4878 #define PTHREAD_CANCEL_DISABLE    1

```

1.8.2. semaphore.h

```

4879
4880 typedef struct
4881 {
4882     struct _pthread_fastlock __sem_lock;
4883     int __sem_value;
4884     _pthread_descr __sem_waiting;
4885 }
4886 sem_t;
4887 #define SEM_FAILED          ((sem_t*)0)
4888
4889 #define SEM_VALUE_MAX       ((int)((~0u)>>1))

```

1.9. Interface Definitions for libpthread

4890 The following interfaces are included in libpthread and are defined by this specification. Unless otherwise noted, these
 4891 interfaces shall be included in the source standard.

4892 Other interfaces listed above for libpthread shall behave as described in the referenced base document.

`_pthread_cleanup_pop`

Name

4893 `_pthread_cleanup_pop` — establish cancellation handlers

Synopsis

```

4894 #include <pthread.h>
4895 | extern void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *, int);

```

Description

4896 ~~Macro~~ The `_pthread_cleanup_pop` ~~defines~~ function provides an implementation of the ~~ABI~~
 4897 `_pthread_cleanup_pop` macro described in *ISO POSIX (2003)*.

4898 The `_pthread_cleanup_pop` function is as ~~specified~~ not in the *Single UNIX Specification, Version 3* source
 4899 standard; it is only in the binary standard.

`_pthread_cleanup_push`

Name

4900 `_pthread_cleanup_push` — establish cancellation handlers

Synopsis

4901 `#include <pthread.h>`
4902 `extern void _pthread_cleanup_push(struct _pthread_cleanup_buffer *, void (*) (void *), void`
4903 `*) ;`

Description

4904 ~~Macro~~The `_pthread_cleanup_push` defines function provides an implementation of the ABI
4905 `_pthread_cleanup_push` macro described in *ISO POSIX (2003)*.
4906 The `_pthread_cleanup_push` function is as specified not in the *Single UNIX Specification, Version 3* source
4907 standard; it is only in the binary standard.

1.10. Interfaces for `libgcc_s`

4908 Table 1-3334 defines the library name and shared object name for the `libgcc_s` library

4909 **Table 1-3334. `libgcc_s` Definition**

Library:	<code>libgcc_s</code>
SONAME:	<code>libgcc_s.so.1</code>

4910

1.10.1. Unwind Library

1.10.1.1. Interfaces for Unwind Library

4911
4912 No external functions are defined for `libgcc_s` - Unwind Library

1.11. Data Definitions for `libgcc_s`

4913 This section defines global identifiers and their values that are associated with interfaces contained in `libgcc_s`. These
4914 definitions are organized into groups that correspond to system headers. This convention is used as a convenience for
4915 the reader, and does not imply the existence of these headers, or their content.
4916 These definitions are intended to supplement those provided in the referenced underlying specifications.
4917 This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are
4918 specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of
4919 these data objects does not preclude their use by other programming languages.

1.11.1. unwind.h

```
4920
4921 struct dwarf_eh_base
4922 {
4923     void *tbase;
4924     void *dbase;
4925     void *func;
4926 }
4927 ;
4928 struct _Unwind_Context;
4929
4930 typedef unsigned int _Unwind_Ptr;
4931 typedef unsigned int _Unwind_Word;
4932
4933 typedef enum
4934 {
4935     _URC_NO_REASON, _URC_FOREIGN_EXCEPTION_CAUGHT = 1, _URC_FATAL_PHASE2_ERROR =
4936     2, _URC_FATAL_PHASE1_ERROR = 3, _URC_NORMAL_STOP = 4, _URC_END_OF_STACK =
4937     5, _URC_HANDLER_FOUND = 6, _URC_INSTALL_CONTEXT =
4938     7, _URC_CONTINUE_UNWIND = 8
4939 }
4940 _Unwind_Reason_Code;
4941
4942 struct _Unwind_Exception
4943 {
4944     _Unwind_Exception_Class;
4945     _Unwind_Exception_Cleanup_Fn;
4946     _Unwind_Word;
4947     _Unwind_Word;
4948 }
4949 ;
4950 #define _UA_SEARCH_PHASE          1
4951 #define _UA_END_OF_STACK          16
4952 #define _UA_CLEANUP_PHASE         2
4953 #define _UA_HANDLER_FRAME         4
4954 #define _UA_FORCE_UNWIND          8
```

1.12. Interfaces for libdl

4955 Table 1-3435 defines the library name and shared object name for the libdl library

4956 **Table 1-3435. libdl Definition**

Library:	libdl
SONAME:	libdl.so.2

4958 The behavior of the interfaces in this library is specified by the following specifications:

- 4959 ~~Linux Standard Base~~this specification
- ~~ISO/IEC 9945: POSIX (2003 Portable Operating System)~~(POSIX)and The Single UNIX® Specification(SUS) V3)

1.12.1. Dynamic Loader

1.12.1.1. Interfaces for Dynamic Loader

An LSB conforming implementation shall provide the generic functions for Dynamic Loader specified in Table 1-3536, with the full functionality as described in the referenced underlying specification.

Table 1-3536. libdl - Dynamic Loader Function Interfaces

dladdr dladdr [1]	dldlclose dclose [2]	dldlerror derror [2]	dlopen dlopen [1]	dlsym dlsym [1]
------------------------------	---------------------------------	---------------------------------	------------------------------	----------------------------

Referenced Specification(s)

- ~~[1]. Linux Standard Base~~this specification
- ~~[2]. ISO/IEC 9945: POSIX (2003 Portable Operating System) and The Single UNIX® Specification (SUS) V3)~~

1.13. Data Definitions for libdl

This section defines global identifiers and their values that are associated with interfaces contained in libdl. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content.

These definitions are intended to supplement those provided in the referenced underlying specifications.

This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are specified in ISO 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.

1.13.1. dlfcn.h

```
#define RTLD_NEXT      ((void *) -11)
#define RTLD_LOCAL     0
#define RTLD_LAZY      0x00001
#define RTLD_NOW       0x00002
#define RTLD_GLOBAL    0x00100

typedef struct
{
    char *dli_fname;
    void *dli_fbase;
    char *dli_sname;
    void *dli_saddr;
}
Dl_info;
```

1.14. Interface Definitions for libdl

4991 The following interfaces are included in libdl and are defined by this specification. Unless otherwise noted, these
 4992 interfaces shall be included in the source standard.

4993 Other interfaces listed above for libdl shall behave as described in the referenced base document.

dladdr

Name

4994 ~~dladdr~~ — library routine for dynamic linking of object files
 4995 dladdr — find the shared object containing a given address

Synopsis

```
4996 #include <dlfcn.h>
4997
4998 typedef struct {
4999     const char *dli_fname;
5000     void       *dli_fbase;
5001     const char *dli_sname;
5002     void       *dli_saddr;
```

```
5003 } Dl_info;
```

```
5004 int dladdr(void *address, Dl_info *dli);
```

Description

```
5005 dladdr implements the System V dynamic linking routines.
```

Return Value

```
5006 dladdr is the inverse of dlsym. If address is successfully located inside a module, dladdr returns a nonzero value,  
5007 otherwise, it returns a 0. On success, dladdr fills in the fields of dli as follows:
```

```
5008 int dladdr(void *addr, Dl_info *dli);
```

Description

```
5009 The dladdr function shall query the dynamic linker for information about the shared object containing the address  
5010 addr. The information shall be returned in the user supplied data structure referenced by dli.
```

```
5011 The structure shall contain at least the following members:
```

```
5012 dli_fname
```

```
5013 — the pathname of the module
```

```
5014 dli_fbase
```

```
5015 — the base address of the module
```

```
5016 dli_sname
```

```
5017 — the name of the highest-addressed symbol whose address precedes the given address
```

```
5018 dli_saddr
```

```
5019 — the address of that symbol
```

```
5020 Shared objects shall be linked using the --shared option to the linker ld(1). The linker flag -rpath may be used to  
5021 add a directory to the default search path for shared objects and shared libraries. The linker flag -E or the C compiler  
5022 flag -rdynamic should be used to cause the application to export its symbols to the shared objects.
```

```
5023 The pathname of the shared object containing the address
```

```
5024 dli_fbase
```

```
5025 The base address at which the shared object is mapped into the address space of the calling process.
```

```
5026 dli_sname
```

```
5027 The name of the nearest runtime symbol with value less than or equal to addr. Where possible, the symbol name  
5028 shall be returned as it would appear in C source code.
```

```
5029 If no symbol with a suitable value is found, both this field and dli_saddr shall be set to NULL.
```

```
5030 dli_saddr
```

5031 The address of the symbol returned in *dli_sname*.
 5032 The behavior of `dladdr` is only specified in dynamically linked programs.

Return Value

5033 On success, `dladdr` shall return non-zero, and the structure referenced by *dli_p* shall be filled in as described.
 5034 Otherwise, `dladdr` shall return zero, and the cause of the error can be fetched with `dlerr`.

Errors

5035 See `dlerr`.

Environment

5036 LD_LIBRARY_PATH
 5037 directory search-path for object files

dlopen

Name

5038 dlopen — open dynamic object

Synopsis

5039 #include <dlfcn.h>
 5040 void * **dlopen**(const char *filename, int flag);

Description

5041 **dlopen** shall behave as specified in ISO/IEC 9945: POSIX (2003-Portable Operating System (POSIX) and The Single
 5042 UNIX® Specification (SUS) V3), but with additional behaviors listed below.

5043 If the file argument does not contain a slash character, then the system shall look for a library of that name in at least
 5044 the following directories, and use the first one which is found:

- 5045 • The directories specified by the DT_RPATH dynamic entry.
- 5046 • The directories specified in the LD_LIBRARY_PATH environment variable (which is a colon separated list of
 5047 pathnames). This step shall be skipped for setuid and setgid executables.
- 5048 • A set of directories sufficient to contain the libraries specified in this standard.[†]

Notes

5050 [†] Traditionally, `/lib` and `/usr/lib`. This case would also cover cases in which the system used the mechanism
 5051 of `/etc/ld.so.conf` and `/etc/ld.so.cache` to provide access.

5052 Example: An application which is not linked against libm may choose to dlopen libm.

dlsym

Name

dlsym — obtain the address of a symbol from a dlopen object

Description

dlsym is as specified in the ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3), but with differences as listed below.

The special purpose value for handle RTLD_NEXT

The value RTLD_NEXT, which is reserved for future use shall be available, with the behavior as described in ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3).

1.15. Interfaces for libcrypt

Table 1-3637 defines the library name and shared object name for the libcrypt library

Table 1-3637. libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

The behavior of the interfaces in this library is specified by the following specifications:

ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.15.1. Encryption

1.15.1.1. Interfaces for Encryption

An LSB conforming implementation shall provide the generic functions for Encryption specified in Table 1-3738, with the full functionality as described in the referenced underlying specification.

Table 1-3738. libcrypt - Encryption Function Interfaces

crypt [1]	encrypt [1]	setkey [1]		
-----------	-------------	------------	--	--

Referenced Specification(s)

[1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.16. Interfaces for libpam

Table 1-3839 defines the library name and shared object name for the libpam library

Table 1-3839. libpam Definition

Library:	libpam
SONAME:	libpam.so.0

A single service name, other, shall always be present. The behavior of this service shall be determined by the system administrator. Additional service names may also exist. ¹⁵

The behavior of the interfaces in this library is specified by the following specifications:

~~Linux Standard Base~~this specification

1.16.1. Pluggable Authentication API

1.16.1.1. Interfaces for Pluggable Authentication API

An LSB conforming implementation shall provide the generic functions for Pluggable Authentication API specified in Table 1-3940, with the full functionality as described in the referenced underlying specification.

Table 1-3940. libpam - Pluggable Authentication API Function Interfaces

pam_acct_mgmt pam_acct_mgmt [1]	pam_close_session pam_close_session [1]	pam_get_item pam_get_item [1]	pam_set_item pam_set_item [1]	pam_strerror pam_strerror [1]
pam_authenticate pam_authenticate [1]	pam_end pam_end [1]	pam_getenvlist pam_getenvlist [1]	pam_setcred pam_setcred [1]	
pam_chautok pam_chautok [1]	pam_fail_delay pam_fail_delay [1]	pam_open_session pam_open_session [1]	pam_start pam_start [1]	

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

1.17. Data Definitions for libpam

This section defines global identifiers and their values that are associated with interfaces contained in libpam. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content.

These definitions are intended to supplement those provided in the referenced underlying specifications.

This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are specified in ISO 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.

1.17.1. security/pam_appl.h

```
typedef struct pam_handle pam_handle_t;
```

```

5095 struct pam_message
5096 {
5097     int msg_style;
5098     const char *msg;
5099 }
5100 ;
5101 struct pam_response
5102 {
5103     char *resp;
5104     int resp_retcode;
5105 }
5106 ;
5107
5108 struct pam_conv
5109 {
5110     int (*conv) (int num_msg, const struct pam_message * *msg,
5111                 struct pam_response * *resp, void *appdata_ptr);
5112     void *appdata_ptr;
5113 }
5114 ;
5115 #define PAM_PROMPT_ECHO_OFF      1
5116 #define PAM_PROMPT_ECHO_ON      2
5117 #define PAM_ERROR_MSG           3
5118 #define PAM_TEXT_INFO           4
5119
5120 #define PAM_SERVICE             1
5121 #define PAM_USER                 2
5122 #define PAM_TTY                 3
5123 #define PAM_RHOST               4
5124 #define PAM_CONV                5
5125 #define PAM_RUSER               8
5126 #define PAM_USER_PROMPT         9
5127
5128 #define PAM_SUCCESS              0
5129 #define PAM_OPEN_ERR            1
5130 #define PAM_USER_UNKNOWN        10
5131 #define PAM_MAXTRIES            11
5132 #define PAM_NEW_AUTHTOK_REQD    12
5133 #define PAM_ACCT_EXPIRED        13
5134 #define PAM_SESSION_ERR        14
5135 #define PAM_CRED_UNAVAIL        15
5136 #define PAM_CRED_EXPIRED        16
5137 #define PAM_CRED_ERR            17
5138 #define PAM_CONV_ERR           19
5139 #define PAM_SYMBOL_ERR          2
5140 #define PAM_AUTHTOK_ERR         20
5141 #define PAM_AUTHTOK_RECOVER_ERR 21
5142 #define PAM_AUTHTOK_LOCK_BUSY   22
5143 #define PAM_AUTHTOK_DISABLE_AGING 23
5144 #define PAM_TRY_AGAIN           24
5145 #define PAM_ABORT               26
5146 #define PAM_AUTHTOK_EXPIRED     27
5147 #define PAM_BAD_ITEM            29

```

```

5148 #define PAM_SERVICE_ERR 3
5149 #define PAM_SYSTEM_ERR 4
5150 #define PAM_BUF_ERR 5
5151 #define PAM_PERM_DENIED 6
5152 #define PAM_AUTH_ERR 7
5153 #define PAM_CRED_INSUFFICIENT 8
5154 #define PAM_AUTHINFO_UNAVAIL 9
5155
5156 #define PAM_DISALLOW_NULL_AUTHTOK 0x0001U
5157 #define PAM_ESTABLISH_CRED 0x0002U
5158 #define PAM_DELETE_CRED 0x0004U
5159 #define PAM_REINITIALIZE_CRED 0x0008U
5160 #define PAM_REFRESH_CRED 0x0010U
5161 #define PAM_CHANGE_EXPIRED_AUTHTOK 0x0020U
5162 #define PAM_SILENT 0x8000U

```

1.18. Interface Definitions for libpam

5163 The following interfaces are included in libpam and are defined by this specification. Unless otherwise noted, these
5164 interfaces shall be included in the source standard.

5165 Other interfaces listed above for libpam shall behave as described in the referenced base document.

pam_acct_mgmt

Name

5166 `pam_acct_mgmt` — establish the status of a user's account

Synopsis

```
5167 #include <security/pam_appl.h>
5168 | extern int pam_acct_mgmt(pam_handle_t *pamh, int flags);
```

Description

5169 `pam_acct_mgmt` establishes the account's usability and the user's accessibility to the system. It is typically called after
5170 the user has been authenticated.

5171 *flags* may be specified as any valid flag (namely, one of those applicable to the *flags* argument of
5172 `pam_authenticate`). Additionally, the value of *flags* may be logically or'd with `PAM_SILENT`.

Return Value

5173 `PAM_SUCCESS`

5174 Success.

5175 `PAM_NEW_AUTHTOK_REQD`

5176 User is valid, but user's authentication token has expired. The correct response to this return-value is to require
5177 that the user satisfy the `pam_chauthtok` function before obtaining service. It may not be possible for an
5178 application to do this. In such a case, the user should be denied access until the account password is updated.

5179 `PAM_ACCT_EXPIRED`

5180 User is no longer permitted access to the system.

5181 `PAM_AUTH_ERR`

5182 Authentication error.

5183 `PAM_PERM_DENIED`

5184 User is not permitted to gain access at this time.

5185 `PAM_USER_UNKNOWN`

5186 User is not known to a module's account management component.

Errors

5187 May be translated to text with `pam_strerror`.

pam_authenticate

Name

5188 pam_authenticate — authenticate the user

Synopsis

```
5189 #include <security/pam_appl.h>
5190 | extern int pam_authenticate(pam_handle_t *pamh, int flags);
```

Description

5191 pam_authenticate serves as an interface to the authentication mechanisms of the loaded modules.

5192 *flags* is an optional parameter that may be specified by the following value:

5193 PAM_DISALLOW_NULL_AUTHTOK

5194 Instruct the authentication modules to return PAM_AUTH_ERR if the user does not have a registered authorization
5195 token.

5196 Additionally, the value of *flags* may be logically or'd with PAM_SILENT.

5197 The process may need to be privileged in order to successfully call this function.

Return Value

5198 PAM_SUCCESS

5199 Success.

5200 PAM_AUTH_ERR

5201 User was not authenticated or process did not have sufficient privileges to perform authentication.

5202 PAM_CRED_INSUFFICIENT

5203 Application does not have sufficient credentials to authenticate the user.

5204 PAM_AUTHINFO_UNAVAIL

5205 Modules were not able to access the authentication information. This might be due to a network or hardware
5206 failure, etc.

5207 PAM_USER_UNKNOWN

5208 Supplied username is not known to the authentication service.

5209 PAM_MAXTRIES

5210 One or more authentication modules has reached its limit of tries authenticating the user. Do not try again.

5211 PAM_ABORT

5212 One or more authentication modules failed to load.

Errors

5213 May be translated to text with `pam_strerror`.

pam_chauthtok

Name

5214 pam_chauthtok — change the authentication token for a given user

Synopsis

```
5215 #include <security/pam_appl.h>
5216 | extern int pam_chauthtok(pam_handle_t *pamh, const int flags);
```

Description

5217 pam_chauthtok is used to change the authentication token for a given user as indicated by the state associated with
5218 the handle *pamh*.

5219 *flags* is an optional parameter that may be specified by the following value:

5220 PAM_CHANGE_EXPIRED_AUTHTOK

5221 User's authentication token should only be changed if it has expired.

5222 Additionally, the value of *flags* may be logically or'd with PAM_SILENT.

RETURN VALUE

5223 PAM_SUCCESS

5224 Success.

5225 PAM_AUTHTOK_ERR

5226 A module was unable to obtain the new authentication token.

5227 PAM_AUTHTOK_RECOVER_ERR

5228 A module was unable to obtain the old authentication token.

5229 PAM_AUTHTOK_LOCK_BUSY

5230 One or more modules were unable to change the authentication token since it is currently locked.

5231 PAM_AUTHTOK_DISABLE_AGING

5232 Authentication token aging has been disabled for at least one of the modules.

5233 PAM_PERM_DENIED

5234 Permission denied.

5235 PAM_TRY_AGAIN

5236 Not all modules were in a position to update the authentication token(s). In such a case, none of the user's
5237 authentication tokens are updated.

5238 PAM_USER_UNKNOWN
 5239 User is not known to the authentication token changing service.

ERRORS

5240 May be translated to text with `pam_strerror`.

pam_close_session

Name

5241 `pam_close_session` — indicate that an authenticated session has ended

Synopsis

```
5242 #include <security/pam_appl.h>
5243 | extern int pam_close_session(pam_handle_t *pamh, int flags);
```

Description

5244 `pam_close_session` is used to indicate that an authenticated session has ended. It is used to inform the module that
 5245 the user is exiting a session. It should be possible for the PAM library to open a session and close the same session
 5246 from different applications.

5247 `flags` may have the value `PAM_SILENT` to indicate that no output should be generated as a result of this function call.

Return Value

5248 PAM_SUCCESS
 5249 Success.

5250 PAM_SESSION_ERR
 5251 One of the required loaded modules was unable to close a session for the user.

Errors

5252 May be translated to text with `pam_strerror`.

pam_end

Name

5253 `pam_end` — terminate the use of the PAM library

Synopsis

```
5254 #include <security/pam_appl.h>
5255 | extern int pam_end(pam_handle_t *pamh, int pam_status);
```

Description

5256 `pam_end` terminates use of the PAM library. On success, the contents of **pamh* are no longer valid, and all memory
 5257 associated with it is invalid.

5258 Normally, *pam_status* is passed the value `PAM_SUCCESS`, but in the event of an unsuccessful service application,
 5259 the appropriate PAM error return value should be used.

Return Value

5260 `PAM_SUCCESS`

5261 Success.

Errors

5262 May be translated to text with `pam_strerror`.

pam_fail_delay

Name

5263 `pam_fail_delay` — specify delay time to use on authentication error

Synopsis

```
5264 #include <security/pam_appl.h>
5265 | extern int pam_fail_delay(pam_handle_t *pamh, unsigned int micro_sec);
```

Description

5266 `pam_fail_delay` specifies the minimum delay for the PAM library to use when an authentication error occurs. The
 5267 actual delay can vary by as much as 25%. If this function is called multiple times, the longest time specified by any of
 5268 the call will be used.

5269 The delay is invoked if an authentication error occurs during the `pam_authenticate` or `pam_chauthtok` function
 5270 calls.

5271 Independent of the success of `pam_authenticate` or `pam_chauthtok`, the delay time is reset to its default value of
 5272 0 when the PAM library returns control to the application from these two functions.

Return Value

5273 `PAM_SUCCESS`

5274 Success.

Errors

5275 May be translated to text with `pam_strerror`.

pam_get_item

Name

5276 `pam_get_item` — obtain the value of the indicated item.

Synopsis

```
5277 #include <security/pam_appl.h>
5278 | extern int pam_get_item(const pam_handle_t *pamh, int item_type, const void **item);
```

Description

5279 `pam_get_item` obtains the value of the indicated *item_type*. The possible values of *item_type* are the same as
5280 listed for `pam_set_item`.

5281 On success, *item* contains a pointer to the value of the corresponding item. Note that this is a pointer to the actual data
5282 and should not be freed or over-written.

Return Value

5283 `PAM_SUCCESS`

5284 Success.

5285 `PAM_PERM_DENIED`

5286 Application passed a `NULL` pointer for *item*.

5287 `PAM_BAD_ITEM`

5288 Application attempted to get an undefined item.

Errors

5289 May be translated to text with `pam_strerror`.

pam_getenvlist

Name

5290 `pam_getenvlist` — returns a pointer to the complete PAM environment.

Synopsis

```
5291 #include <security/pam_appl.h>
5292 | extern char * const *pam_getenvlist(pam_handle_t *pamh);
```

Description

5293 `pam_getenvlist` returns a pointer to the complete PAM environment. This pointer points to an array of pointers to
 5294 NUL-terminated strings and must be terminated by a `NULL` pointer. Each string has the form "name=value".
 5295 The PAM library module allocates memory for the returned value and the associated strings. The calling application is
 5296 responsible for freeing this memory.

Return Value

5297 `pam_getenvlist` returns an array of string pointers containing the PAM environment. On error, `NULL` is returned.

pam_open_session

Name

5298 `pam_open_session` — used to indicate that an authenticated session has been initiated

Synopsis

```
5299 #include <security/pam_appl.h>
5300 | extern int pam_open_session(pam_handle_t *pamh, int flags);
```

Description

5301 `pam_handle_t` is used to indicate that an authenticated session has begun. It is used to inform the module that the
 5302 user is currently in a session. It should be possible for the PAM library to open a session and close the same session
 5303 from different applications.

5304 *flags* may have the value `PAM_SILENT` to indicate that no output be generated as a result of this function call.

Return Value

5305 `PAM_SUCCESS`

5306 Success.

5307 `PAM_SESSION_ERR`

5308 One of the loaded modules was unable to open a session for the user.

ERRORS

5309 May be translated to text with `pam_strerror`.

pam_set_item

Name

5310 `pam_set_item` — (re)set the value of an item.

Synopsis

```
5311 #include <security/pam_appl.h>
5312 | extern int pam_set_item(pam_handle_t *pamh, int item_type, const void *item);
```

Description

5313 `pam_set_item` (re)sets the value of one of the following `item_types`:

5314 `PAM_SERVICE`

5315 service name

5316 `PAM_USER`

5317 user name

5318 `PAM_TTY`

5319 terminal name

5320 The value for a device file should include the `/dev/` prefix. The value for graphical, X-based, applications should
5321 be the `$DISPLAY` variable.

5322 `PAM_RHOST`

5323 remote host name

5324 `PAM_CONV`

5325 conversation structure

5326 `PAM_RUSER`

5327 remote user name

5328 `PAM_USER_PROMPT`

5329 string to be used when prompting for a user's name

5330 The default value for this string is `Please enter username: .`

5331 For all *item_types* other than `PAM_CONV`, *item* is a pointer to a NULL-terminated character string. In the case of
5332 `PAM_CONV`, *item* points to an initialized `pam_conv` structure.

Return Value

5333 `PAM_SUCCESS`

5334 Success.

- 5335 PAM_PERM_DENIED
- 5336 An attempt was made to replace the conversation structure with a `NULL` value.
- 5337 PAM_BUF_ERR
- 5338 Function ran out of memory making a copy of the item.
- 5339 PAM_BAD_ITEM
- 5340 Application attempted to set an undefined item.

Errors

- 5341 May be translated to text with `pam_strerror`.

pam_setcred

Name

5342 `pam_setcred` — set the module-specific credentials of the user

Synopsis

```
5343 #include <security/pam_appl.h>
5344 extern int pam_setcred(pam_handle_t *pamh, int flags);
```

Description

5345 `pam_setcred` sets the module-specific credentials of the user. It is usually called after the user has been authenticated,
 5346 after the account management function has been called and after a session has been opened for the user.

5347 *flags* maybe specified from among the following values:

5348 `PAM_ESTABLISH_CRED`

5349 set credentials for the authentication service

5350 `PAM_DELETE_CRED`

5351 delete credentials associated with the authentication service

5352 `PAM_REINITIALIZE_CRED`

5353 reinitialize the user credentials

5354 `PAM_REFRESH_CRED`

5355 extend lifetime of the user credentials

5356 Additionally, the value of *flags* may be logically or'd with `PAM_SILENT`.

Return Value

5357 `PAM_SUCCESS`

5358 Success.

5359 `PAM_CRED_UNAVAIL`

5360 Module cannot retrieve the user's credentials.

5361 `PAM_CRED_EXPIRED`

5362 User's credentials have expired.

5363 `PAM_USER_UNKNOWN`

5364 User is not known to an authentication module.

5365 `PAM_CRED_ERR`

5366 Module was unable to set the credentials of the user.

Errors

5367 May be translated to text with `pam_strerror`.

pam_start

Name

5368 `pam_start` — initialize the PAM library

Synopsis

```
5369 #include <security/pam_appl.h>
5370 | extern int pam_start(const char *service_name, const char *user, const (struct pam_conv
5371   *pam_conversation), pam_handle_t **pamh);
```

Description

5372 `pam_start` is used to initialize the PAM library. It must be called prior to any other usage of the PAM library. On
 5373 success, **pamh* becomes a handle that provides continuity for successive calls to the PAM library. `pam_start`
 5374 expects arguments as follows: the *service_name* of the program, the *username* of the individual to be
 5375 authenticated, a pointer to an application-supplied `pam_conv` structure, and a pointer to a *pam_handle_t* pointer.

5376 An application must provide the *conversation function* used for direct communication between a loaded module and
 5377 the application. The application also typically provides a means for the module to prompt the user for a password, etc.

5378 The structure, `pam_conv`, is defined to be,

```
5379     struct pam_conv {
5380         int (*conv) (int num_msg,
5381                     const struct pam_message * *msg,
5382                     struct pam_response * *resp,
5383                     void *appdata_ptr);
5384     void *appdata_ptr;
```

5385 };

5386 It is initialized by the application before it is passed to the library. The contents of this structure are attached to the
 5387 **pamh* handle. The point of this argument is to provide a mechanism for any loaded module to interact directly with
 5388 the application program; this is why it is called a conversation structure.

5389 When a module calls the referenced *conv* function, *appdata_ptr* is set to the second element of this structure.

5390 The other arguments of a call to *conv* concern the information exchanged by module and application. *num_msg* holds
 5391 the length of the array of pointers passed via *msg*. On success, the pointer *resp* points to an array of *num_msg*
 5392 *pam_response* structures, holding the application-supplied text. Note that *resp* is a struct *pam_response* array and not
 5393 an array of pointers.

Return Value

5394 PAM_SUCCESS

5395 Success.

5396 PAM_BUF_ERR

5397 Memory allocation error.

5398 PAM_ABORT

5399 Internal failure.

ERRORS

5400 May be translated to text with *pam_strerror*.

pam_strerror

Name

5401 *pam_strerror* — returns a string describing the PAM error

Synopsis

5402 #include <security/pam_appl.h>
 5403 | ~~extern~~ const char * **pam_strerror**(*pam_handle_t* **pamh*, int *errnum*);

Description

5404 *pam_strerror* returns a string describing the PAM error associated with *errnum*.

Return Value

5405 On success, this function returns a description of the indicated error. The application should not free or modify this
 5406 string. This returned string will not be translated.

Notes

1. The LSB generally does not include interfaces unlikely to be used by software applications.
1. As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a kernel patch would be accepted if submitted.
1. For example, if `off_t` is 64 bits.
1. As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a kernel patch would be accepted if submitted.
1. As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a kernel patch would be accepted if submitted.
1. `SIOCGIFCONF` is similar to the `if_nameindex` family found in the *Single UNIX Specification, Version 3* or the `getifaddrs` family found in BSD.
2. Historical UNIX systems disagree on the meaning of the return value.
1. This was a deliberate Linus decision after an unpopular experiment in including the calling process in the 2.5.1 kernel. See "What does it mean to signal everybody?", Linux Weekly News, 20 December 2001, <http://lwn.net/2001/1220/kernel.php3>
1. As of spring 2004, we don't know of any Linux kernel patches to switch to ENXIO, but we believe that such a kernel patch would be accepted if submitted.
1. Note the optional use of the buffer, unlike the `strerror_r` found in the *Single UNIX Specification, Version 3*, in which the message is always copied into the supplied buffer. The return types also differ.
1. A token is a nonempty string of characters not occurring in the string `delim`, followed by `\0` or by a character occurring in `delim`.
1. The Linux kernel has deliberately chosen `EISDIR` for this case and does not expect to change (Al Viro, personal communication).
1. These macros take the `stat` buffer (an `int`) as an argument—not a pointer to the buffer!
1. Traditionally, `/lib` and `/usr/lib`. This case would also cover cases in which the system used the mechanism of `/etc/ld.so.conf` and `/etc/ld.so.cache` to provide access.
Example: An application which is not linked against `libm` may choose to `dlopen libm`.
15. Future versions of this specification might define additional service names.

II. Utility Libraries

Chapter 2. utility Libraries

An LSB-conforming implementation may also support some utility libraries which are built on top of the interfaces provided by the base libraries. These libraries implement common functionality, and hide additional system dependent information such as file formats and device names.

2.1. Interfaces for libz

Table 2-1 defines the library name and shared object name for the libz library

Table 2-1. libz Definition

Library:	libz
SONAME:	libz.so.1

The behavior of the interfaces in this library is specified by the following specifications:

zlib 1.2 Manual

2.1.1. Compression Library

2.1.1.1. Interfaces for Compression Library

An LSB conforming implementation shall provide the generic functions for Compression Library specified in Table 2-2, with the full functionality as described in the referenced underlying specification.

Table 2-2. libz - Compression Library Function Interfaces

adler32adler32 [1]	deflateInit_deflateInit_ [1]	gzerrorgzerror [1]	gzreadgzread [1]	inflateInit2_inflateInit2_ [1]
compresscompress [1]	deflateParamsdeflateParams [1]	gzflushgzflush [1]	gzrewindgzrewind [1]	inflateInit_inflateInit_ [1]
compress2compress2 [1]	deflateResetdeflateReset [1]	gzgetcgzgetc [1]	gzseekgzseek [1]	inflateResetinflateReset [1]
crc32crc32 [1]	deflateSetDictionarydeflateSetDictionary [1]	gzgetcsgzgetc [1]	gzsetparamsgzsetparams [1]	inflateSetDictionaryinflateSetDictionary [1]
deflatedeflate [1]	get_crc_tableget_crc_table [1]	gzopengzopen [1]	gztellgztell [1]	inflateSyncinflateSync [1]
deflateCopydeflateCopy [1]	gzclosegzclose [1]	gzprintfgzprintf [1]	gzwritegzwrite [1]	inflateSyncPointinflateSyncPoint [1]
deflateEnddeflateEnd [1]	gzdopengzdopen [1]	gzputcgzputc [1]	inflateinflate [1]	uncompressuncompress [1]

<code>deflateInit2_deflateInit2_[1]</code>	<code>gzEOFgzEOF [1]</code>	<code>gzputsgzputs [1]</code>	<code>inflateEndinflateEnd [1]</code>	<code>zErrorzError [1]</code>
--	-----------------------------	-------------------------------	---------------------------------------	-------------------------------

Referenced Specification(s)

[1]. zlib 1.2-Manual

2.2. Data Definitions for libz

This section defines global identifiers and their values that are associated with interfaces contained in libz. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content.

These definitions are intended to supplement those provided in the referenced underlying specifications.

This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are specified in ISO 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.

2.2.1. zlib.h

```

#define Z_NULL 0
#define MAX_WBITS 15
#define MAX_MEM_LEVEL 9
#define deflateInit2(strm, level, method, windowBits, memLevel, strategy)
deflateInit2_((strm), (level), (method), (windowBits), (memLevel), (strategy), ZLIB_VERSION,
sizeof(z_stream))
#define deflateInit(strm, level) deflateInit_((strm), (level), ZLIB_VERSION,
sizeof(z_stream))
#define inflateInit2(strm, windowBits) inflateInit2_((strm), (windowBits), ZLIB_VERSION,
sizeof(z_stream))
#define inflateInit(strm) inflateInit_((strm), ZLIB_VERSION, sizeof(z_stream))

typedef int intf;

typedef void *voidpf;
typedef unsigned int uInt;
typedef unsigned long uLong;
typedef uLong uLongf;
typedef void *voidp;
typedef unsigned char Byte;
typedef off_t z_off_t;
typedef void *const voidpc;

typedef voidpf (*alloc_func) (voidpf opaque, uInt items, uInt size);
typedef void (*free_func) (voidpf opaque, voidpf address);
struct internal_state
{
    int dummy;
}
;
```

```

54  typedef Byte Bytef;
55  typedef uInt uIntf;
56
57  typedef struct z_stream_s
58  {
59      Bytef *next_in;
60      uInt avail_in;
61      uLong total_in;
62      Bytef *next_out;
63      uInt avail_out;
64      uLong total_out;
65      char *msg;
66      struct internal_state *state;
67      alloc_func zalloc;
68      free_func zfree;
69      voidpf opaque;
70      int data_type;
71      uLong Adler;
72      uLong reserved;
73  }
74  z_stream;
75
76  typedef z_stream *z_streamp;
77  typedef voidp gzFile;
78  #define Z_NO_FLUSH      0
79  #define Z_PARTIAL_FLUSH 1
80  #define Z_SYNC_FLUSH    2
81  #define Z_FULL_FLUSH    3
82  #define Z_FINISH        4
83
84  #define Z_ERRNO (-1)
85  #define Z_STREAM_ERROR (-2)
86  #define Z_DATA_ERROR   (-3)
87  #define Z_MEM_ERROR     (-4)
88  #define Z_BUF_ERROR     (-5)
89  #define Z_OK            0
90  #define Z_STREAM_END    1
91  #define Z_NEED_DICT     2
92
93  #define Z_DEFAULT_COMPRESSION (-1)
94  #define Z_NO_COMPRESSION      0
95  #define Z_BEST_SPEED          1
96  #define Z_BEST_COMPRESSION    9
97
98  #define Z_DEFAULT_STRATEGY 0
99  #define Z_FILTERED          1
100 #define Z_HUFFMAN_ONLY      2
101
102 #define Z_BINARY             0
103 #define Z_ASCII              1
104 #define Z_UNKNOWN            2
105
106 #define Z_DEFLATED           8

```

2.3. Interfaces for libncurses

Table 2-3 defines the library name and shared object name for the libncurses library

Table 2-3. libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

The behavior of the interfaces in this library is specified by the following specifications:

CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018

2.3.1. Curses

2.3.1.1. Interfaces for Curses

An LSB conforming implementation shall provide the generic functions for Curses specified in Table 2-4, with the full functionality as described in the referenced underlying specification.

Table 2-4. libncurses - Curses Function Interfaces

addch addch [1]	has_ has_ic [1]	mvwaddchnstr mvwaddchnstr [1]	ser_init scr_init [1]	vwscanw vwscanw [1]
addchnstr addchnstr [1]	has_ has_il [1]	mvwaddchstr mvwaddchstr [1]	ser_restore scr_restore [1]	waddch waddch [1]
addchnstr addchstr [1]	hline hline [1]	mvwaddnstr mvwaddnstr [1]	ser_setscr scr_set [1]	waddchnstr waddchnstr [1]
addnstr addnstr [1]	ideok idlok [1]	mvwaddstr mvwaddstr [1]	ser_scr scr [1]	waddchstr waddchstr [1]
addstr addstr [1]	idlok idlok [1]	mvwchgat mvwchgat [1]	ser_rollback scr [1]	waddnstr waddnstr [1]
attr_get attr_get [1]	immedok immedok [1]	mvwdeletch mvwdeletch [1]	ser_rollback scr [1]	waddstr waddstr [1]
attr_off attr_off [1]	inch inch [1]	mvwgetch mvwgetch [1]	set_curt termset_curt [1]	wattr_get wattr_get [1]
attr_on attr_on [1]	inchstr inchstr [1]	mvwgetnstr mvwgetnstr [1]	set_term termset_term [1]	wattr_off wattr_off [1]
attr_set attr_set [1]	inchstr inchstr [1]	mvwgetstr mvwgetstr [1]	setscrreg scrreg [1]	wattr_on wattr_on [1]
attroff attroff [1]	init_color init_color [1]	mvwhline mvwhline [1]	setupterm setupterm [1]	wattr_set wattr_set [1]

<code>attronattron</code> [1]	<code>init_pairinit_pair</code> [1]	<code>mvwinmvwin</code> [1]	<code>slk_attr_setslk_attr_set</code> [1]	<code>wattroffwattroff</code> [1]
<code>attrsetattrset</code> [1]	<code>initserinitscr</code> [1]	<code>mvwinchmvwinch</code> [1]	<code>slk_attr_offslk_attr_of</code> [1]	<code>wattronwattron</code> [1]
<code>baudratebaudrate</code> [1]	<code>inistrinnstr</code> [1]	<code>mvwinchnstrmvwinchnstr</code> [1]	<code>slk_attronslk_attron</code> [1]	<code>wattrsetwattrset</code> [1]
<code>beepbeep</code> [1]	<code>inschinsch</code> [1]	<code>mvwinchstrmvwinchstr</code> [1]	<code>slk_attrsetslk_attrset</code> [1]	<code>wbkgdwbkgd</code> [1]
<code>bkgdwbkgd</code> [1]	<code>insdellinsdelln</code> [1]	<code>mvwinwnstrmvwinwnstr</code> [1]	<code>slk_clearslk_clear</code> [1]	<code>wbkgdsetwbkgdset</code> [1]
<code>bkgdsetwbkgdset</code> [1]	<code>insertlninsertln</code> [1]	<code>mvwinschmvwinsch</code> [1]	<code>slk_colorslk_color</code> [1]	<code>wborderwborder</code> [1]
<code>borderborder</code> [1]	<code>insnstrinsnstr</code> [1]	<code>mvwinsnstrmvwinsnstr</code> [1]	<code>slk_initslk_init</code> [1]	<code>wchgatwchgat</code> [1]
<code>boxbox</code> [1]	<code>insstrinsstr</code> [1]	<code>mvwinsstrmvwinsstr</code> [1]	<code>slk_labelslk_label</code> [1]	<code>wclearwclear</code> [1]
<code>can_change_colorcan_change_color</code> [1]	<code>instrinstr</code> [1]	<code>mvwinstrmvwinstr</code> [1]	<code>slk_noutrefreshslk_noutrefresh</code> [1]	<code>wclrtobotwclrtobot</code> [1]
<code>cbreakcbreak</code> [1]	<code>intrflushintrflush</code> [1]	<code>mvwprintwmvwprintw</code> [1]	<code>slk_refreshslk_refresh</code> [1]	<code>wclrtoeolwclrtoeol</code> [1]
<code>chgatchgat</code> [1]	<code>is_linetouchedis_linetouched</code> [1]	<code>mvwscanwmvwscanw</code> [1]	<code>slk_restoreslk_restore</code> [1]	<code>wcolor_setwcolor_set</code> [1]
<code>clearclear</code> [1]	<code>is_wintouchedis_wintouched</code> [1]	<code>mvwvlinemvwvline</code> [1]	<code>slk_setslk_set</code> [1]	<code>wcursyncupwcursyncup</code> [1]
<code>clearokclearok</code> [1]	<code>isendwinisendwin</code> [1]	<code>napmsnapms</code> [1]	<code>slk_touchslk_touch</code> [1]	<code>wdelehwdelehw</code> [1]
<code>clrtobotclrtobot</code> [1]	<code>keynamekeyname</code> [1]	<code>newpadnewpad</code> [1]	<code>standendstandend</code> [1]	<code>wdeletelnwdeleteln</code> [1]
<code>clrtoeolclrtoeol</code> [1]	<code>keypadkeypad</code> [1]	<code>newtermnewterm</code> [1]	<code>standoutstandout</code> [1]	<code>wechocharwechochar</code> [1]
<code>color_contentcolor_content</code> [1]	<code>killcharkillchar</code> [1]	<code>newwinnewwin</code> [1]	<code>start_colorstart_color</code> [1]	<code>werasewerase</code> [1]
<code>color_setcolor_set</code> [1]	<code>leaveokleaveok</code> [1]	<code>nlnl</code> [1]	<code>subpadsubpad</code> [1]	<code>wgetchwgetch</code> [1]
<code>copywincopywin</code> [1]	<code>longnamelongname</code> [1]	<code>nocbreaknocbreak</code> [1]	<code>subwinsubwin</code> [1]	<code>wgetnstrwgetnstr</code> [1]
<code>curs_setcurs_set</code> [1]	<code>metameta</code> [1]	<code>nodelaynodelay</code> [1]	<code>syncoksyncok</code> [1]	<code>wgetstrwgetstr</code> [1]

<code>def_prog_mode</code> <code>def_prog_mode [1]</code>	<code>move</code> <code>move [1]</code>	<code>noecho</code> <code>noecho [1]</code>	<code>termattr</code> <code>termattr [1]</code>	<code>whline</code> <code>whline [1]</code>
<code>def_shell_mode</code> <code>def_shell_mode [1]</code>	<code>mvaddch</code> <code>mvaddch [1]</code>	<code>nonl</code> <code>nonl [1]</code>	<code>termname</code> <code>termname [1]</code>	<code>winch</code> <code>winch [1]</code>
<code>del_curterm</code> <code>del_curterm [1]</code>	<code>mvaddchstr</code> <code>mvaddchstr [1]</code>	<code>noqiflush</code> <code>noqiflush [1]</code>	<code>tgetent</code> <code>tgetent [1]</code>	<code>winchnstr</code> <code>winchnstr [1]</code>
<code>delay_output</code> <code>delay_output [1]</code>	<code>mvaddchstr</code> <code>mvaddchstr [1]</code>	<code>noraw</code> <code>noraw [1]</code>	<code>tgetflag</code> <code>tgetflag [1]</code>	<code>winchstr</code> <code>winchstr [1]</code>
<code>delch</code> <code>delch [1]</code>	<code>mvaddnstr</code> <code>mvaddnstr [1]</code>	<code>notimeout</code> <code>notimeout [1]</code>	<code>tgetnum</code> <code>tgetnum [1]</code>	<code>winnstr</code> <code>winnstr [1]</code>
<code>delete</code> <code>delete [1]</code>	<code>mvaddstr</code> <code>mvaddstr [1]</code>	<code>overlay</code> <code>overlay [1]</code>	<code>tgetstr</code> <code>tgetstr [1]</code>	<code>winsch</code> <code>winsch [1]</code>
<code>delscreen</code> <code>delscreen [1]</code>	<code>mvchgat</code> <code>mvchgat [1]</code>	<code>overwrite</code> <code>overwrite [1]</code>	<code>tgoto</code> <code>tgoto [1]</code>	<code>winsdell</code> <code>winsdell [1]</code>
<code>delwin</code> <code>delwin [1]</code>	<code>mvcur</code> <code>mvcur [1]</code>	<code>pair_content</code> <code>pair_content [1]</code>	<code>tigetflag</code> <code>tigetflag [1]</code>	<code>winsertln</code> <code>winsertln [1]</code>
<code>derwin</code> <code>derwin [1]</code>	<code>mvdelch</code> <code>mvdelch [1]</code>	<code>pechochar</code> <code>pechochar [1]</code>	<code>tigetnum</code> <code>tigetnum [1]</code>	<code>winsnstr</code> <code>winsnstr [1]</code>
<code>doupdate</code> <code>doupdate [1]</code>	<code>mvderwin</code> <code>mvderwin [1]</code>	<code>pnoutrefresh</code> <code>pnoutrefresh [1]</code>	<code>tigetstr</code> <code>tigetstr [1]</code>	<code>winsstr</code> <code>winsstr [1]</code>
<code>dupwin</code> <code>dupwin [1]</code>	<code>mvgetch</code> <code>mvgetch [1]</code>	<code>prefresh</code> <code>prefresh [1]</code>	<code>timeout</code> <code>timeout [1]</code>	<code>winstr</code> <code>winstr [1]</code>
<code>echo</code> <code>echo [1]</code>	<code>mvgetnstr</code> <code>mvgetnstr [1]</code>	<code>printw</code> <code>printw [1]</code>	<code>touchline</code> <code>touchline [1]</code>	<code>wmove</code> <code>wmove [1]</code>
<code>echochar</code> <code>echochar [1]</code>	<code>mvgetstr</code> <code>mvgetstr [1]</code>	<code>putp</code> <code>putp [1]</code>	<code>touchwin</code> <code>touchwin [1]</code>	<code>wnoutrefresh</code> <code>wnoutrefresh [1]</code>
<code>endwin</code> <code>endwin [1]</code>	<code>mvhline</code> <code>mvhline [1]</code>	<code>putwin</code> <code>putwin [1]</code>	<code>tparm</code> <code>tparm [1]</code>	<code>wprintw</code> <code>wprintw [1]</code>
<code>erase</code> <code>erase [1]</code>	<code>mvinch</code> <code>mvinch [1]</code>	<code>qiflush</code> <code>qiflush [1]</code>	<code>tputs</code> <code>tputs [1]</code>	<code>wredrawln</code> <code>wredrawln [1]</code>
<code>erasechar</code> <code>erasechar [1]</code>	<code>mvinchstr</code> <code>mvinchstr [1]</code>	<code>raw</code> <code>raw [1]</code>	<code>typeahead</code> <code>typeahead [1]</code>	<code>wrefresh</code> <code>wrefresh [1]</code>
<code>filter</code> <code>filter [1]</code>	<code>mvinchstr</code> <code>mvinchstr [1]</code>	<code>redrawwin</code> <code>redrawwin [1]</code>	<code>unctrl</code> <code>unctrl [1]</code>	<code>wscanw</code> <code>wscanw [1]</code>
<code>flash</code> <code>flash [1]</code>	<code>mvinnstr</code> <code>mvinnstr [1]</code>	<code>refresh</code> <code>refresh [1]</code>	<code>ungetch</code> <code>ungetch [1]</code>	<code>wscr</code> <code>wscr [1]</code>
<code>flushinp</code> <code>flushinp [1]</code>	<code>mvinsch</code> <code>mvinsch [1]</code>	<code>reset_prog_mode</code> <code>reset_prog_mode [1]</code>	<code>untouchwin</code> <code>untouchwin [1]</code>	<code>wsetscreg</code> <code>wsetscreg [1]</code>
<code>getbkgd</code> <code>getbkgd [1]</code>	<code>mvinsnstr</code> <code>mvinsnstr [1]</code>	<code>reset_shell_mode</code> <code>reset_shell_mode [1]</code>	<code>use_env</code> <code>use_env [1]</code>	<code>wstandend</code> <code>wstandend [1]</code>

	[1]	et_shell_mode [1]		d [1]
getch getch [1]	mvinsstr mvinsstr [1]	resetty resetty [1]	vidattr vidattr [1]	wstandout wstandout [1]
getnstr getnstr [1]	mvinstr mvinstr [1]	restartterm restartterm [1]	vidputs vidputs [1]	wsyncdown wsyncdown [1]
getstr getstr [1]	mvprintw mvprintw [1]	riponline riponline [1]	vline vline [1]	wsyncup wsyncup [1]
getwin getwin [1]	mvscanw mvscanw [1]	savetty savetty [1]	vw_printw vw_printw [1]	wtimeout wtimeout [1]
halfdelay halfdelay [1]	mvvline mvvline [1]	scanw scanw [1]	vw_scanw vw_scanw [1]	wtouchln wtouchln [1]
has_colors has_colors [1]	mvwaddch mvwaddch [1]	scr_dump scr_dump [1]	vwprintw vwprintw [1]	wvline wvline [1]

Referenced Specification(s)

[1]. CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018

An LSB conforming implementation shall provide the generic data interfaces for Curses specified in Table 2-5, with the full functionality as described in the referenced underlying specification.

Table 2-5. libncurses - Curses Data Interfaces

COLORS COLORS [1]	COLS COLS [1]	acs_map acs_map [1]	curscr curscr [1]	
COLOR_PAIRS COLOR_PAIRS [1]	LINES LINES [1]	cur_term cur_term [1]	stdscr stdscr [1]	

Referenced Specification(s)

[1]. CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018

2.4. Data Definitions for libncurses

This section defines global identifiers and their values that are associated with interfaces contained in libncurses.

These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content.

These definitions are intended to supplement those provided in the referenced underlying specifications.

This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are specified in ISO 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.

2.4.1. curses.h

```

134
135 #define ERR      (-1)
136 #define OK       (0)
137 #define ACS_RARROW      (acs_map[ '+' ])
138 #define ACS_LARROW      (acs_map[ ', ' ])
139 #define ACS_UARROW      (acs_map[ '- ' ])
140 #define ACS_DARROW      (acs_map[ '. ' ])
141 #define ACS_BLOCK       (acs_map[ '0' ])
142 #define ACS_CKBOARD     (acs_map[ 'a' ])
143 #define ACS_DEGREE      (acs_map[ 'f' ])
144 #define ACS_PLMINUS     (acs_map[ 'g' ])
145 #define ACS_BOARD       (acs_map[ 'h' ])
146 #define ACS_LANTERN     (acs_map[ 'i' ])
147 #define ACS_LRCORNER    (acs_map[ 'j' ])
148 #define ACS_URCORNER    (acs_map[ 'k' ])
149 #define ACS_ULCORNER    (acs_map[ 'l' ])
150 #define ACS_LLCORNER    (acs_map[ 'm' ])
151 #define ACS_PLUS        (acs_map[ 'n' ])
152 #define ACS_S1 (acs_map[ 'o' ])
153 #define ACS_HLINE       (acs_map[ 'q' ])
154 #define ACS_S9 (acs_map[ 's' ])
155 #define ACS_LTEE        (acs_map[ 't' ])
156 #define ACS_RTEE        (acs_map[ 'u' ])
157 #define ACS_BTEE        (acs_map[ 'v' ])
158 #define ACS_TTEE        (acs_map[ 'w' ])
159 #define ACS_VLINE       (acs_map[ 'x' ])
160 #define ACS_DIAMOND      (acs_map[ '` ' ])
161 #define ACS_BULLET      (acs_map[ '~ ' ])
162 #define getmaxyx(win,y,x)
163 (y=(win)?((win)->_maxy+1):ERR,x=(win)?((win)->_maxx+1):ERR)
164 #define getbegyx(win,y,x) (y=(win)?(win)->_begy:ERR,x=(win)?(win)->_begx:ERR)
165 #define getyx(win,y,x) (y=(win)?(win)->_cury:ERR,x=(win)?(win)->_curx:ERR)
166 #define getparyx(win,y,x) (y=(win)?(win)->_pary:ERR,x=(win)?(win)->_parx:ERR)
167
168 #define WA_ALTCHARSET    A_ALTCHARSET
169 #define WA_ATTRIBUTES    A_ATTRIBUTES
170 #define WA_BLINK         A_BLINK
171 #define WA_BOLD A_BOLD
172 #define WA_DIM A_DIM
173 #define WA_HORIZONTAL    A_HORIZONTAL
174 #define WA_INVIS         A_INVIS
175 #define WA_LEFT A_LEFT
176 #define WA_LOW A_LOW
177 #define WA_NORMAL        A_NORMAL
178 #define WA_PROTECT       A_PROTECT
179 #define WA_REVERSE       A_REVERSE
180 #define WA_RIGHT         A_RIGHT
181 #define WA_STANDOUT       A_STANDOUT
182 #define WA_TOP A_TOP
183 #define WA_UNDERLINE     A_UNDERLINE

```

```

184  #define WA_VERTICAL      A_VERTICAL
185  #define A_REVERSE        NCURSES_BITS(1UL,10)
186
187  #define COLOR_BLACK      0
188  #define COLOR_RED        1
189  #define COLOR_GREEN      2
190  #define COLOR_YELLOW     3
191  #define COLOR_BLUE       4
192  #define COLOR_MAGENTA    5
193  #define COLOR_CYAN       6
194  #define COLOR_WHITE      7
195
196  #define _SUBWIN 0x01
197  #define _ENDLINE 0x02
198  #define _FULLWIN 0x04
199  #define _ISPAD 0x10
200  #define _HASMOVED 0x20
201
202  typedef unsigned char bool;
203
204  typedef unsigned long chtype;
205  typedef struct screen SCREEN;
206  typedef struct _win_st WINDOW;
207  typedef chtype attr_t;
208  typedef struct
209  {
210      attr_t attr;
211      wchar_t chars[5];
212  }
213  cchar_t;
214  struct pdat
215  {
216      short _pad_y;
217      short _pad_x;
218      short _pad_top;
219      short _pad_left;
220      short _pad_bottom;
221      short _pad_right;
222  }
223  ;
224
225  struct _win_st
226  {
227      short _cury;
228      short _curx;
229      short _maxy;
230      short _maxx;
231      short _begy;
232      short _begx;
233      short _flags;
234      attr_t _attrs;
235      chtype _bkgd;
236      bool _notimeout;

```

```

237     bool _clear;
238     bool _leaveok;
239     bool _scroll;
240     bool _idlok;
241     bool _idcok;
242     bool _immed;
243     bool _sync;
244     bool _use_keypad;
245     int _delay;
246     struct ldat *_line;
247     short _regtop;
248     short _regbottom;
249     int _parx;
250     int _pary;
251     WINDOW *_parent;
252     struct pdat _pad;
253     short _yoffset;
254     cchar_t _bkgrnd;
255 }
256 ;
257 #define KEY_CODE_YES      0400
258 #define KEY_BREAK        0401
259 #define KEY_MIN           0401
260 #define KEY_DOWN         0402
261 #define KEY_UP           0403
262 #define KEY_LEFT         0404
263 #define KEY_RIGHT        0405
264 #define KEY_HOME         0406
265 #define KEY_BACKSPACE    0407
266 #define KEY_F0           0410
267 #define KEY_DL           0510
268 #define KEY_IL           0511
269 #define KEY_DC           0512
270 #define KEY_IC           0513
271 #define KEY_EIC          0514
272 #define KEY_CLEAR        0515
273 #define KEY_EOS          0516
274 #define KEY_EOL          0517
275 #define KEY_SF           0520
276 #define KEY_SR           0521
277 #define KEY_NPAGE        0522
278 #define KEY_PPAGE        0523
279 #define KEY_STAB         0524
280 #define KEY_CTAB         0525
281 #define KEY_CATAB        0526
282 #define KEY_ENTER        0527
283 #define KEY_SRESET       0530
284 #define KEY_RESET        0531
285 #define KEY_PRINT        0532
286 #define KEY_LL           0533
287 #define KEY_A1           0534
288 #define KEY_A3           0535
289 #define KEY_B2           0536

```

```

290  #define KEY_C1  0537
291  #define KEY_C3  0540
292  #define KEY_BTAB      0541
293  #define KEY_BEG 0542
294  #define KEY_CANCEL    0543
295  #define KEY_CLOSE     0544
296  #define KEY_COMMAND   0545
297  #define KEY_COPY      0546
298  #define KEY_CREATE    0547
299  #define KEY_END 0550
300  #define KEY_EXIT      0551
301  #define KEY_FIND      0552
302  #define KEY_HELP      0553
303  #define KEY_MARK      0554
304  #define KEY_MESSAGE    0555
305  #define KEY_MOVE      0556
306  #define KEY_NEXT      0557
307  #define KEY_OPEN      0560
308  #define KEY_OPTIONS    0561
309  #define KEY_PREVIOUS   0562
310  #define KEY_REDO       0563
311  #define KEY_REFERENCE  0564
312  #define KEY_REFRESH    0565
313  #define KEY_REPLACE    0566
314  #define KEY_RESTART    0567
315  #define KEY_RESUME     0570
316  #define KEY_SAVE       0571
317  #define KEY_SBEG       0572
318  #define KEY_SCANCEL    0573
319  #define KEY_SCOMMAND   0574
320  #define KEY_SCOPY      0575
321  #define KEY_SCREATE    0576
322  #define KEY_SDC 0577
323  #define KEY_SDL 0600
324  #define KEY_SELECT     0601
325  #define KEY_SEND       0602
326  #define KEY_SEOL       0603
327  #define KEY_SEXIT      0604
328  #define KEY_SFIND      0605
329  #define KEY_SHELP      0606
330  #define KEY_SHOME      0607
331  #define KEY_SIC 0610
332  #define KEY_SLEFT      0611
333  #define KEY_SMESSAGE    0612
334  #define KEY_SMOVE      0613
335  #define KEY_SNEXT      0614
336  #define KEY_SOPTIONS    0615
337  #define KEY_SPREVIOUS   0616
338  #define KEY_SPRINT      0617
339  #define KEY_SREDO       0620
340  #define KEY_SREPLACE    0621
341  #define KEY_SRIGHT     0622
342  #define KEY_SRSUME     0623

```

```

343 #define KEY_SSAVE      0624
344 #define KEY_SSUSPEND  0625
345 #define KEY_SUNDO      0626
346 #define KEY_SUSPEND    0627
347 #define KEY_UNDO       0630
348 #define KEY_MOUSE      0631
349 #define KEY_RESIZE     0632
350 #define KEY_MAX 0777
351
352 #define PAIR_NUMBER(a) (((a)& A_COLOR)>>8)
353 #define NCURSES_BITS(mask,shift) ((mask)<<((shift)+8))
354 #define A_CHARTEXT      (NCURSES_BITS(1UL,0)-1UL)
355 #define A_NORMAL        0L
356 #define NCURSES_ATTR_SHIFT 8
357 #define A_COLOR NCURSES_BITS(((1UL)<<8)-1UL,0)
358 #define A_BLINK NCURSES_BITS(1UL,11)
359 #define A_DIM NCURSES_BITS(1UL,12)
360 #define A_BOLD NCURSES_BITS(1UL,13)
361 #define A_ALTCHARSET NCURSES_BITS(1UL,14)
362 #define A_INVIS NCURSES_BITS(1UL,15)
363 #define A_PROTECT NCURSES_BITS(1UL,16)
364 #define A_HORIZONTAL NCURSES_BITS(1UL,17)
365 #define A_LEFT NCURSES_BITS(1UL,18)
366 #define A_LOW NCURSES_BITS(1UL,19)
367 #define A_RIGHT NCURSES_BITS(1UL,20)
368 #define A_TOP NCURSES_BITS(1UL,21)
369 #define A_VERTICAL NCURSES_BITS(1UL,22)
370 #define A_STANDOUT NCURSES_BITS(1UL,8)
371 #define A_UNDERLINE NCURSES_BITS(1UL,9)
372 #define COLOR_PAIR(n) NCURSES_BITS(n,0)
373 #define A_ATTRIBUTES NCURSES_BITS(~(1UL-1UL),0)

```

2.5. Interfaces for libutil

374 Table 2-6 defines the library name and shared object name for the libutil library

375 **Table 2-6. libutil Definition**

Library:	libutil
SONAME:	libutil.so.1

377 The behavior of the interfaces in this library is specified by the following specifications:

378 ~~Linux Standard Base~~ this specification

2.5.1. Utility Functions

379 2.5.1.1. Interfaces for Utility Functions

380 An LSB conforming implementation shall provide the generic functions for Utility Functions specified in Table 2-7,
381 with the full functionality as described in the referenced underlying specification.

Table 2-7. libutil - Utility Functions Function Interfaces

forkpty forkpty [1]	login_tty login_tty [1]	logwtmp logwtmp [1]		
login login [1]	logout logout [1]	openpty openpty [1]		

Referenced Specification(s)

[1]. ~~Linux Standard Base~~this specification

2.6. Interface Definitions for libutil

The following interfaces are included in libutil and are defined by this specification. Unless otherwise noted, these interfaces shall be included in the source standard.

Other interfaces listed above for libutil shall behave as described in the referenced base document.

forkpty

Name

~~forkpty — find and open~~Create a new process attached to an available pseudo-~~tty~~terminal

Synopsis

```
#include <pty.h>

int forkpty(int * amaster,
            _____char * name,
            _____ struct termios * termp,
```

```
394 | _____ struct winsize * winp);
```

Description

395 | The `forkpty()` function joins shall find and open a pseudo-terminal device pair in the same manner as the
 396 | `openpty()`, `fork()`, and `login_tty()` to () function. If a pseudo-terminal is available, `forkpty` shall create a new
 397 | process operating on a pseudo-tty. The file descriptor of the master side of the pseudo-tty is returned in `amaster`, and
 398 | null or the filename of the slave in `name`. If non null, the same manner as the `fork()` function, and prepares the new
 399 | process for login in the same manner as `login_tty()`.

400 | If `term` and is not null, it shall refer to a `termios` structure that shall be used to initialize the characteristics of the
 401 | slave device. If `winp` parameters will determine the terminal attributes and is not null, it shall refer to a `winsize`
 402 | structure used to initialize the window size of the slave side of the pseudo-tty device.

Return Value

403 | On success of the child process, zero is returned. When, the parent process receives shall return the PID process id of
 404 | its the child, and the child process, pid is returned shall return 0. On error, no new process shall be created, -1 is shall be
 405 | returned, and `errno` is shall be set appropriately. On success, the parent process shall receive the file descriptor of the
 406 | master side of the pseudo-terminal in the location referenced by `amaster`, and, if `name` is not NULL, the filename of
 407 | the slave device in `name`.

Errors

408 | EAGAIN

409 | Unable to create a new process.

410 | ENOENT

411 | There are no available pseudo-terminals.

412 | ENOMEM

413 | Insufficient memory was available.

login

Name

414 login — login utility function

Synopsis

```
415 #include <utmp.h>
416 void login (struct utmp * ut );
```

Description

417 The `login()` function ~~updates~~ shall update the `/var/run/utmp` and `/var/log/wtmp` files with user information contained
418 in accounting databases. The `ut` parameter shall reference a `utmp` structure for all fields except the following:

- 419 1. The `ut_type` field shall be set to `USER_PROCESS`.
- 420 2. The `ut_pid` field shall be set to the process identifier for the current process.
- 421 3. The `ut_line` field shall be set to the name of the controlling terminal device. The name shall be found by
422 examining the device associated with the standard input, output and error streams in sequence, until one
423 associated with a terminal device is found. If none of these streams refers to a terminal device, the `ut_line` field
424 shall be set to "???". If the terminal device is in the `/dev` directory hierarchy, the `ut_line` field shall not
425 contain the leading `"/dev/"`, otherwise it shall be set to the final component of the pathname of the device. If the
426 user accounting database imposes a limit on the size of the `ut_line` field, it shall truncate the name, but any
427 such limit shall not be smaller than `UT_LINESIZE` (including a terminating null character).

Return Value

428 None

Errors

429 None

login_tty

Name

430 `login_tty` — ~~find and open an available pseudo-tty~~ Prepare a terminal for login

Synopsis

```
431 #include <utmp.h>
432 int login_tty (int fd);
```

Description

433 ~~login_tty() sets up for a login on~~ The `login_tty()` function shall prepare the `ty` terminal device referenced by the file
 434 descriptor `fd`. This function ~~creates~~ shall create a new session, makes the `ty` for the current process terminal the
 435 controlling terminal, ~~sets~~ for the current process, and set the standard input, output, and error streams of the current
 436 process, ~~and closes fd~~.

Return Value

437 ~~On success, zero~~ to the terminal. If `fd` is ~~returned~~. ~~On~~ not the standard input, output or ~~error~~, stream, then
 438 `login_tty()` shall close `fd`.

Return Value

439 On success, `login_tty()` shall return zero; otherwise -1 is returned, and `errno` is shall be set appropriately.

Errors

440 ENOTTY
 441 `fd` does not refer to a terminal device.

logout

Name

442 `logout` — `logout` utility function

Synopsis

```
443    #include <utmp.h>
444    int logout (const char * line );
```

Description

445 Given the device *line*, the `logout()` function ~~removes~~ shall search the user accounting database which is read by
 446 `getutent` for an entry from with the corresponding ~~`/var/run/utmp` system file~~ *line*, and with the type of
 447 `USER_PROCESS`. If a corresponding entry is located, it shall be updated as follows:

- 448 1. The `ut_name` field shall be set to zeroes (`UT_NAMESIZE` NUL bytes).
- 449 2. The `ut_host` field shall be set to zeroes (`UT_HOSTSIZE` NUL bytes).
- 450 3. The `ut_tv` shall be set to the current time of day.
- 451 4. The `ut_type` field shall be set to `DEAD_PROCESS`.

Return Value

452 On success, the `logout()` function shall return non-zero. Zero is returned if there was no entry to remove. ~~A non-zero~~
 453 ~~return value indicates success~~, or if the `utmp` file could not be opened or updated.

logwtmp

Name

454 `logwtmp` — append an entry to the `wtmp` file

Synopsis

```
455    #include <utmp.h>
456    |
```

```
457 void logwtmp(const char *line, const char *name, const char *host);
```

Description

```
458 logwtmp() constructs an utmp structure using line, name, host, current time and current process id. Then it calls
459 updwtmp() to append the structure to the utmp file.
```

Availability

```
460 Both functions are available under glibc2, but not under libc5. However, logwtmp occurs in the old libbsd.
```

Files

```
461 /var/log/wtmp database of past user logins
462 void logwtmp (const char * line , const char * name , const char * host );
```

Description

```
463 If the process has permission to update the user accounting databases, the logwtmp function shall append a record to
464 the user accounting database that records all logins and logouts. The record to be appended shall be constructed as
465 follows:
```

- 466 1. The `ut_line` field shall be initialized from `line`. If the user accounting database imposes a limit on the size of
467 the `ut_line` field, it shall truncate the value, but any such limit shall not be smaller than `UT_LINESIZE`
468 (including a terminating null character).
- 469 2. The `ut_name` field shall be initialized from `name`. If the user accounting database imposes a limit on the size of
470 the `ut_name` field, it shall truncate the value, but any such limit shall not be smaller than `UT_NAMESIZE`
471 (including a terminating null character).
- 472 3. The `ut_host` field shall be initialized from `host`. If the user accounting database imposes a limit on the size of
473 the `ut_host` field, it shall truncate the value, but any such limit shall not be smaller than `UT_HOSTSIZE`
474 (including a terminating null character).
- 475 4. If the `name` parameter does not refer to an empty string (i.e. `" "`), the `ut_type` field shall be set to
476 `USER_PROCESS`; otherwise the `ut_type` field shall be set to `DEAD_PROCESS`.
- 477 5. The `ut_id` field shall be set to the process identifier for the current process.
- 478 6. The `ut_tv` field shall be set to the current time of day.

```
479 If a process does not have write access to the the user accounting database, the logwtmp function will not update
480 it. Since the function does not return any value, an application has no way of knowing whether it succeeded or
481 failed.
```

Return Value

```
482 None.
```

openpty

Name

483 `openpty` — find and open an available pseudo-`ty`terminal

Synopsis

```
484 #include <pty.h>

485 int openpty(int *amaster,
486             _____int *aslave,
487             _____char *name,
488             _____struct termios *termp,
489             _____struct winsize *winp);
```

Description

490 The `openpty()` function ~~finds~~ shall find an available pseudo-`ty`terminal and returns file descriptors for the master
 491 and slave devices in the locations referenced by *amaster* and *aslave*. The ~~respective~~. If *name* is not NULL, the
 492 filename of the slave ~~is returned in name, otherwise a null. The terminal parameters of the slave will~~ shall be set to the
 493 ~~values placed in the user supplied buffer referenced by name. If termp, otherwise a null. The window size of the~~
 494 ~~slave will be set~~ is not NULL, it shall point to a `termios` structure used to initialize the ~~values in~~ terminal parameters
 495 of the slave pseudo-terminal device. If *winp*, ~~otherwise a null~~ is not NULL, it shall point to a `winsize` structure used
 496 to initialize the window size parameters of the slave pseudo-terminal device.

Return Value

497 On success, zero is returned. On error, -1 is returned, and `errno` is set appropriately.

Errors

498 ENOENT
 499 There are no available ~~ty~~s pseudo-terminals.

III. Commands and Utilities

Chapter 3. Commands and Utilities

3.1. Commands and Utilities

If any operand (except one which follows —) starts with a hyphen the behavior is unspecified.¹

The following table lists the Commands and Utilities. Unless otherwise specified the command or utility is described in the Single UNIX Specification (SUS). When an interface is not defined in the Single UNIX Specification, then the next prevailing standard is referenced (ie., POSIX, SVID).

The behavior of the interfaces described in this section are specified by the following standards.

Table 3-1 lists the Commands and Utilities required to be present on a conforming system. These commands and utilities shall behave as described in the relevant underlying specification, with the following exceptions:

1. If any operand (except one which follows --) starts with a hyphen, the behavior is unspecified.

Rationale (Informative)

Applications should place options before operands, or use --, as needed. This text is needed because GNU option parsing differs from POSIX. For example, **ls . -a** in GNU **ls** means to list the current directory, showing all files (that is, "." is an operand and -a is an option). In POSIX, "." and -a are both operands, and the command means to list the current directory, and also the file named -a. Suggesting that applications rely on the setting of the POSIXLY_CORRECT environment variable, or try to set it, seems worse than just asking the applications to invoke commands in ways which work with either the POSIX or GNU behaviors.

The behavior of the interfaces described in this section is specified by the following standards.

Linux Standard Base²this specification

ISO/IEC 9945:2003 Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3³ISO
POSIX (2003)

Table 3-1. Commands and Utilities

[³ [1]	ar ² [2]	at ² [2]	awk ² [2]	basename ³ [1]
batch ² [2]	bc ² [2]	cat ³ [1]	chfn ² [2]	chgrp ² [2]
chmod ³ [1]	chown ² [2]	chsh ² [2]	cksum ³ [1]	cmp ³ [1]
col ² [2]	comm ³ [1]	cp ³ [1]	cpio ² [2]	crontab ² [2]
csplit ³ [1]	cut ² [2]	date ³ [1]	dd ³ [1]	df ² [2]
diff ³ [1]	dirname ³ [1]	dmesg ² [2]	du ² [2]	echo ² [2]
egrep ² [2]	env ³ [1]	expand ³ [1]	expr ³ [1]	false ³ [1]
fgrep ² [2]	file ² [2]	find ² [2]	fold ³ [1]	fuser ² [2]
gencat ³ [1]	getconf ³ [1]	gettext ² [2]	grep ² [2]	groupadd ² [2]
groupdel ² [2]	groupmod ² [2]	groups ² [2]	gunzip ² [2]	gzip ² [2]

head ³ [1]	hostname ² [2]	iconv ³ [1]	id ³ [1]	install ² [2]
install_initd ² [2]	ipcrm ² [2]	ipcs ² [2]	join ³ [1]	kill ³ [1]
killall ² [2]	ln ³ [1]	locale ³ [1]	localedef ³ [1]	logname ³ [1]
lpr ² [2]	ls ² [2]	lsb_release ² [2]	m4 ² [2]	make ³ [1]
man ³ [1]	md5sum ² [2]	mkdir ³ [1]	mkfifo ² [1]	mknod ² [2]
mktemp ² [2]	more ² [2]	mount ² [2]	msgfmt ² [2]	mv ³ [1]
newgrp ² [2]	nice ³ [1]	nl ³ [1]	nohup ³ [1]	od ² [2]
passwd ² [2]	paste ³ [1]	patch ² [2]	pathchk ³ [1]	pidof ² [2]
pr ³ [1]	printf ³ [1]	ps ³ [1]	pwd ³ [1]	remove_initd ² [2]
renice ² [2]	rm ³ [1]	rmdir ³ [1]	sed ² [2]	sendmail ² [2]
sh ³ [1]	shutdown ² [2]	sleep ³ [1]	sort ² [1]	split ³ [1]
strip ³ [1]	stty ³ [1]	su ² [2]	sync ² [2]	tail ³ [1]
tar ² [2]	tee ³ [1]	test ³ [1]	time ³ [1]	touch ³ [1]
tr ³ [1]	true ³ [1]	tsort ³ [1]	tty ³ [1]	umount ² [2]
uname ³ [1]	unexpand ³ [1]	uniq ³ [1]	useradd ² [2]	userdel ² [2]
usermod ² [2]	wc ³ [1]	xargs ² [2]		

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

3.2. Command Behavior

This section contains descriptions for commands and utilities whose specified behavior in the LSB contradicts or extends the standards referenced. It also contains commands and utilities only required by the LSB and not specified by other standards.

ar

Name

26 ar — create and maintain library archives (LSB DEPRECATED)

Description

27 ar is deprecated from the LSB and is expected to disappear from a future version of the LSB. ⁺

Rationale

29 The LSB generally does not include software development utilities nor does it specify .o and .a file formats.

30 ar is as specified in the Single UNIX Specification ISO POSIX (2003) but with differences as listed below.

Differences

31 -T

32 -C

33 need not be accepted.

34 -l

35 has unspecified behavior.

36 -q

37 has unspecified behavior; using -r is suggested.

Notes

39 1. ~~The LSB generally does not include software development utilities nor does it specify .o and .a file formats.~~

at

Name

40 at — examine or delete jobs for later execution

Description

41 | **at** is as specified in the Single UNIX Specification ISO POSIX (2003) but with differences as listed below.

Differences

42 -d

43 | is functionally equivalent to the -r option specified in the Single UNIX Specification ISO POSIX (2003).

44 -r

45 need not be supported, but the '-d' option is equivalent.

46 -t time

47 need not be supported.

Files

48 The files at.allow and at.deny reside in /etc rather than /usr/lib/cron.

awk

Name

49 awk — pattern scanning and processing language

Description

50 | **awk** is as specified in the Single UNIX Specification ISO POSIX (2003) but with differences as listed below.

Differences

51 Certain aspects of internationalized regular expressions are optional; see Internationalization and Regular
52 Expressions>.

batch

Name

53 | `batch` — ~~executes~~ schedule commands when the system load permits to be executed in a batch queue

Description

54 | The specification for **batch** is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with the following
55 | differences as listed below.

Files

56 |
57 | The files `at.allow` and `at.deny` reside in `/etc` rather than `/usr/lib/cron`.

bc

Name

58 | `bc` — An arbitrary precision calculator language

Description

59 | **bc** is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003) but with differences as listed below.

Differences

60 | The `bc` language may be extended in an implementation defined manner. If an implementation supports extensions, it
61 | shall also support the additional options:

62 | `-s, --standard`

63 | processes exactly the POSIX `bc` language.

64 | `-w, --warn`

65 | gives warnings for extensions to POSIX `bc`.

chfn

Name

66 `chfn` — change user name and information

Synopsis

67 `chfn [-f full_name] [-h home_phone] [user]`

Description

68 ~~**chfn** changes user fullname and other information for a user's account. This information is typically printed by finger~~
 69 ~~and similar programs. A normal user may only change the fields for their own account, the super user may change the~~
 70 ~~fields for any account.~~

71 ~~The only restrictions placed on the contents of the fields is that no control characters may be present, nor any of~~
 72 ~~comma, colon, or equal sign.~~

73 `chfn [-f full_name] [-h home_phone] [user]`

Description

74 **chfn** shall update the user database. An unprivileged user may only change the fields for their own account, a user with
 75 appropriate privileges may change the fields for any account.

76 The fields *full_name* and *home_phone* may contain any character except:

any control character
 comma
 colon
 equal sign

If none of the options are selected, **chfn** operates in an interactive fashion. The prompts and expected input in interactive mode are unspecified and should not be relied upon.

As it is possible for the system to be configured to restrict which fields a non-privileged user is permitted to change, applications should be written to gracefully handle these situations.

Standard Options

-f full_name
 sets the user's full name.

-h home_phone
 sets the user's home phone number.

+

Notes

Future Directions

The following two options are expected to be added in a future version of the LSB:

-o office
 sets the user's office room number.

-p office_phone
 sets the user's office phone number.

Note that some implementations contain a "-o other" option which specifies an additional field called "other". Traditionally, this field is not subject to the constraints about legitimate characters in fields. Also, one traditionally shall have appropriate privileges to change the other field. At this point there is no consensus about whether it is desirable to specify the other field; applications may wish to avoid using it.

The "-w work_phone" field found in some implementations should be replaced by the "-p office_phone" field. The "-r room_number" field found in some implementations is the equivalent of the "-o office" option mentioned above; which one of these two options to specify will depend on implementation experience and the decision regarding the other field.

~~The intention is for chfn to match the behavior of finger; some historical implementations have been broken in the sense that finger and chfn do not agree on what the fields are.~~

chgrp

Name

103 chgrp — change file group

Description

104 **chgrp** is as specified in ~~the Single UNIX Specification~~ISO POSIX (2003) but with differences as listed below.

Differences

105 The -L, -H, and -P options need not be supported.

chown

Name

106 chown — change file owner and group

Description

107 **chown** is as specified in ~~the Single UNIX Specification~~ISO POSIX (2003) but with differences as listed below.

Differences

108 The -L, -H, and -P options need not be supported.

chsh

Name

109 `chsh` — change login shell

Synopsis

110 `chsh [-s login_shell] [user]`

Description

111 **chsh** changes the user login shell. This determines the name of the user's initial login command. ~~A normal~~An
 112 unprivileged user may only change the login shell for their own account, the super user with appropriate privilege
 113 may change the login shell for any account- specified by *user*.

114 ~~The only restrictions placed on~~Unless the user has appropriate privilege, the initial login shell is that the command
 115 name shall be one of those listed in `/etc/shells`, unless the invoker is the super user, and then any value may. The
 116 *login_shell* shall be added the absolute path (i.e. it must start with '/') to an executable file. Accounts which are
 117 restricted (in an implementation-defined manner) may not change their login shell.

118 If the `-s` option is not selected, **chsh** operates in an interactive mode. The prompts and expected input in this mode are
 119 ~~implementation-defined~~unspecified.

Standard Options

120 `-s login_shell`
 121 sets the login shell.

col

Name

122 `col` — filter reverse line feeds from input

Description

123 **col** is as specified in the ~~The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5~~
 124 ~~(ISBN: 1-85912-191-8, C604)~~SUSv2 with the difference that the **-p** option has unspecified behavior.

125 Although **col** is shown as legacy in the Single UNIX SpecificationSUSv2, Version 2, it is not (yet) deprecated in the
 126 LSB.

cpio

Name

127 `cpio` — copy file archives in and out

Description

128 **cpio** is as specified in the Single UNIX Specification ISO POSIX (2003), but with differences as listed below.

Differences

129 Certain aspects of internationalized filename globbing are optional; see Some elements of the Pattern Matching
130 Notation are optional; see Internationalization and Filename Globbing Pattern Matching Notation>.

crontab

Name

131 `crontab` — maintain crontab files for individual users

Synopsis

132 **crontab** [`--u user`] `file`
133 **crontab** [`--u user`] `{-}` `{-l | -r | -e}`

Description

134 **crontab** is as specified in the Single UNIX Specification ISO POSIX (2003), but with differences as listed below.

Files

135 The files `cron.allow` and `cron.deny` reside in `/etc` rather than `/usr/lib/cron`.

cut

Name

136 `cut` — split a file into sections determined by context lines

Description

137 `cut` is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

Differences

138 `-n`

139 has unspecified behavior.

df

Name

140 `df` — report filesystem disk space usage

Description

141 `df` is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with the following differences.

142 If the `-k` option is not specified, disk space is shown in unspecified units. Applications should specify `-k`.

143 If an argument is the absolute file name of a disk device node containing a mounted filesystem, `df` shows the space
144 available on that filesystem rather than on the filesystem containing the device node (which is always the root
145 filesystem).

dmesg

Name

```
146 | dmesg — print or control the kernel ringbuffer system message buffer
```

Synopsis

```
147 | dmesg [--c +--| -n level +--| -s bufsize-]
```

Description

148 **dmesg** examines or controls the kernel ring-buffer system message buffer. Only a user with appropriate privileges may
149 modify the system message buffer parameters or contents.

Standard Options

150	$-C$
-----	------

151	If the user has appropriate privilege, clears the ring system message buffer contents after printing.
-----	---

152 *-n level*

153	If the user has appropriate privilege, sets the level at which logging of messages is done to the console.
-----	--

```
154  -s bufsize
```

uses a buffer of *bufsize* to query the kernel-ringsysmsg message buffer. This is 819616392 by default (this matches the default kernel syslog buffer size in 2.0.33 and since 2.1.403113). If you have set the kernel buffer to larger than the default then this option can be used to view the entire buffer.

du

Name

158 du — estimate file space usage

Description

159 **du** is as specified in the ~~Single UNIX Specification~~ISO POSIX (2003), but with differences as listed below.

Differences

160 If the `-k` option is not specified, disk space is shown in unspecified units. Applications should specify `-k`.

echo

Name

161 `echo` — display a line of text

Synopsis

162 `echo` [STRING+...]

Description

163 The **echo** command is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with the following
 164 differences.

165 Unlike the behavior specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), whether **echo** supports options is
 166 implementation defined. The behavior of **echo** if any arguments contain backslashes is also implementation defined.

167 ~~Applications~~ Conforming applications shall not run **echo** with a first argument starting with a hyphen, or with any
 168 arguments containing backslashes; they shall use **printf** in those cases. ⁺

Notes

170 ~~1.~~ The behavior specified here is similar to that specified by ~~the Single UNIX Specification version 3~~ ISO POSIX
 171 (2003) without the XSI option. However, the LSB forbids all options and the latter forbids only `-n`.

egrep

Name

172 `egrep` — search a file with an ERE pattern

Description

173 **egrep** is equivalent to `grep -E`. For further details, see the specification for **grep**.

fgrep

Name

174 `fgrep` — search a file with a fixed pattern

Description

175 **fgrep** is equivalent to `grep -F`. For further details, see the specification for **grep**.

file

Name

176 `file` — determine file type

Description

177 **file** is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

Differences

178 The `-M`, `-h`, `-d`, and `-i` options need not be supported.

find

Name

179 `find` — search for files in a directory hierarchy

Description

180 **find** is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with additional options as specified
181 below.

Differences

182 ~~Certain aspects of internationalized filename globbing are optional; see~~ Some elements of the Pattern Matching
183 ~~Notation are optional; see Internationalization and Filename Globbing~~ Pattern Matching Notation>.

fuser

Name

184 `fuser` — identify processes using files or sockets

Description

185 `fuser` is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

Differences

186 `-c`

187 has unspecified behavior.

188 `-f`

189 has unspecified behavior.

gettext

Name

190 `gettext` — retrieve text string from message database catalog

Synopsis

191 `gettext` [`-options`—]`[—]` [`textdomain`—] `msgid`

192

193 | **gettext** -s [-options-] msgid-----

Description

194 The **gettext** utility retrieves a translated text string corresponding to string *msgid* from a message object generated
 195 with **msgfmt** utility.

196 The message object name is derived from the optional argument *textdomain* if present, otherwise from the
 197 | TEXTDOMAIN environment variable. If no domain is specified, or if a corresponding string cannot be found, **gettext**
 198 prints *msgid*.

199 Ordinarily **gettext** looks for its message object in *dirname/lang/LC_MESSAGES* where *dirname* is the
 200 implementation-defined default directory and *lang* is the locale name. If present, the TEXTDOMAINDIR environment
 201 variable replaces the *dirname*.

202 This utility interprets C escape sequences such as \t for tab. Use \\ to print a backslash. To produce a message on a
 203 | line of its own, either put a \n at the end of *msgid*, or use this command in conjunction with the **printf** utility.

204 When used with the -s option the **gettext** utility behaves like the **echo** utility. ~~But it does not simply copy its~~
 205 ~~arguments, except that the message corresponding to standard output. Instead those messages found~~*msgid* in the
 206 | selected catalog ~~are translated~~ provides the arguments.

Options

207 -d *domainname*
 208 --domain=*domainname*

209 | RetrievePARAMETER translated messages from domainname.

210 -e

211 | Enable expansion of some escape sequences.

212 -n

213 | Suppress trailing newline.

Operands

214 The following operands are supported:

215 *textdomain*

216 | A domain name used to retrieve the messages.

217 *msgid*

218 | A key to retrieve the localized message.

Environment Variables

219 LANGUAGE

220 | Specifies one or more locale names. ~~See gettext message handling functions for more information.~~

221 LANG
 222 Specifies locale name.

223 LC_MESSAGES
 224 Specifies messaging locale, and if present overrides LANG for messages.

225 TEXTDOMAIN
 226 Specifies the text domain name, which is identical to the message object filename without .mo suffix.

227 TEXTDOMAINDIR
 228 Specifies the pathname to the message database catalog, and if present replaces the implementation-defined
 229 default directory.

Exit Status

230 The following exit values are returned:

231 0
 232 Successful completion.

233 >0
 234 An error occurred.

grep

Name

235 `grep` — print lines matching a pattern

Description

236 **grep** is as specified in the Single UNIX Specification but with differences as listed below.

LSB Differences

237 Certain aspects of internationalized regular expressions are optional; see Internationalization and Regular
 238 Expressions ISO POSIX (2003) >=,
 239 , but with differences as listed below.

LSB Differences

240 Some elements of the Pattern Matching Notation are optional; see Internationalization and Pattern Matching Notation.

groupadd

Name

241 groupadd — create a new group

Synopsis

242 **groupadd** [-g *gid* [-o]] *group*

Description

243 If the caller has appropriate privilege, the **groupadd** command shall create a new group named *group*. The group
 244 name shall be unique in the group database. If no *gid* is specified, **groupadd** shall create the new group with a unique
 245 group ID.

Options

246 -g *gid* [-o]

247 — specifies the numerical value of the group's ID. This value shall be unique, unless the -o option is used. The value
 248 shall be non-negative.

249 The new group shall have group ID *gid*. If the -o option is not used, no other group shall have this group ID.

250 The value of *gid* shall be non-negative.

groupdel

Name

251 groupdel — delete a group

Synopsis

252 **groupdel** *group*

Description

253 If the caller has sufficient privilege, the **groupdel** command shall modify the system group database, deleting the
 254 group named *group*. If the group named *group* does not exist, **groupdel** shall issue a diagnostic message and exit
 255 with a non-zero exit status.

groupmod

Name

256 groupmod — modify a group

Synopsis

257 **groupmod** [-g *gid* [-o]] [-n *group_name*] *group*

Description

258 ~~groupdel~~ modifies the system account files, deleting all entries that refer to *group*. The named *group* shall exist.

groupmod

Name

259 ~~groupmod~~ — modify a group

Synopsis

260 ~~groupmod~~ [-g *gid* [-o]] [-n *group_name*] *group*

261 If the caller has appropriate privilege, the **groupmod** command shall modify the entry in the system group database
262 corresponding to a group named *group*.

Options

263 -g *gid* [-o]

264 ~~specifies the numerical value of~~ Modify the group's ID. This value shall be unique, unlessgroup ID, setting it to
265 *gid*. If the -o option is not used, no other group shall have this group ID. The value of *gid*shall be non-negative.
266 ~~Any files which~~

267 Only the group ID in the old-group-ID database is altered; any files with group ownership set to the file-group
268 ID shall have the fileoriginal group ID changed manuallyare unchanged by this modification.

269 -n *group_name*

270 changes the name of the group from *group* to *group_name*.

groups

Name

271 `groups` — display a group

Synopsis

272 `groups` [*user*]

Description

273 The **groups** displays the current group ID names or values. If the value does not have a corresponding entry command
 274 shall behave as **id -Gn [*user*]**, as specified in the group database, the value will be displayed as the numerical group
 275 value ISO POSIX (2003). The optional *user* parameter will display the groups for the named user.

gunzip

Name

276 `gunzip` — uncompress files

Description

277 **gunzip** is equivalent to **gzip -d**. See the specification for **gzip** for further details.

gzip

Name

278 `gzip` — compress or expand files

Synopsis

279 `gzip` [~~—acd~~fhlLnNrtvV19~~—~~] [~~-S~~ suffix] [~~-name~~—...—...]

Description

280 The **gzip** ~~tries~~command shall attempt to reduce the size of the named files. Whenever possible, each file is replaced by
 281 one with the extension `.gz`, while keeping the same ownership modes, access and modification times. If no files are
 282 specified, or if a file name is `"-"`, the standard input is compressed to the standard output. **gzip** ~~will~~shall only attempt
 283 to compress regular files. In particular, it will ignore symbolic links.

284 When compressing, `gzip` uses the deflate algorithm specified in ~~RFC1951~~RFC 1951: DEFLATE Compressed Data
 285 Format Specification and stores the result in a file using the `gzip` file format specified in ~~RFC1952~~RFC 1952: GZIP
 286 File Format Specification.

Options

287 `-a, --ascii`

288 does nothing on ~~Linux~~LSB conforming systems.

289 This option may be deprecated in a future verion of this specification.

290 `-c, --stdout, --to-stdout`

291 writes output on standard output; ~~keeps~~, leaving the original files unchanged. If there are several input files, the
 292 output consists of a sequence of independently compressed members. To obtain better compression, concatenate
 293 all input files before compressing them.

294 `-d, --decompress, --uncompress`

295 ~~—decompresses.~~

296 the name operands are compressed files, and **gzip** shall decompress them.

297 `-f, --force`

298 forces compression or decompression even if the file has multiple links or the corresponding file already exists,
 299 or if the compressed data is read from or written to a terminal. If the input data is not in a format recognized by
 300 **gzip**, and if the option `--stdout` is also given, copy the input data without change to the standard ouput: let
 301 **gzip** behave as **cat**. If `-f` is not given, and when not running in the background, **gzip** prompts to verify whether
 302 an existing file should be overwritten.

303 `-l, --list`

304 lists the compressed size, uncompressed size, ratio and uncompressed name for each compressed file. Gives the
 305 uncompressed size as -1 for files not in **gzip** format. Additionally displays method, crc and timestamp for the
 306 uncompress file when used in combination with `--verbose`.

307 ~~The~~For decompression, **gzip** shall support at least the following compression methods ~~currently supported are~~:

- 308 • ~~deflate~~, (RFC 1951: DEFLATE Compressed Data Format Specification)
- 309 • ~~compress~~, (ISO POSIX (2003))
- 310 • ~~lzh~~ (SCO **compress -H**) ~~and~~
- 311 • ~~pack~~, (Huffman encoding)

312 The crc ~~is~~shall be given as `ffffffff` for a file not in **gzip** format.

313 With `--name`, the uncompressed name, date and time are those stored within the compress file, if present.

314 With `--verbose`, the size totals and compression ratio for all files is also displayed, unless some sizes are
 315 unknown. With `--quiet`, the title and totals lines are not displayed.

316 **-L, --license**

317 displays the **gzip** license and quit.

318 **-n, --no-name**

319 does not save the original file name and time stamp by default when compressing. (The original name is always
 320 saved if the name had to be truncated.) When decompressing, do not restore the original file name if present
 321 (remove only the gzip suffix from the compressed file name) and do not restore the original time stamp if present
 322 (copy it from the compressed file). This option is the default when decompressing.

323 **-N, --name**

324 always saves the original file name and time stamp when compressing; this is the default. When decompressing,
 325 restore the original file name and time stamp if present. This option is useful on systems which have a limit on file
 326 name length or when the time stamp has been lost after a file transfer.

327 **-q, --quiet**

328 suppresses all warnings.

329 **-r, --recursive**

330 travels the directory structure recursively. If any of the file names specified on the command line are directories,
 331 **gzip** will descend into the directory and compress all the files it finds there (or decompress them in the case of
 332 **gunzip**).

333 **-S .suf, --suffix .suf**

334 uses suffix `.suf` instead of `.gz`.

335 **-t, --test**

336 checks the compressed file integrity.

337 **-v, --verbose**

338 displays the name and percentage reduction for each file compressed or decompressed.

```

339  -#, --fast, --best
340      regulates the speed of compression using the specified digit #, where -1 or --fast indicates the fastest
341      compression method (less compression) and -9 or --best indicates the slowest compression method (best
342      compression). The default compression level is -6 (that is, biased towards high compression at expense of
343      speed).

```

LSB Deprecated Options

```

344  The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should
345  only use the non-LSB-deprecated behaviors.

```

```

346  -V, --version
347      displays the version number and compilation options, then quits.

```

hostname

Name

```

348  hostname — show or set the system's host name

```

Synopsis

```

349  hostname [-v] [-a] [--alias] [-d] [--domain] [-f] [--fqdn]
350  [-i] [--ip address] [-l long] [-s] [--short] [-y] [--yp]
351  [--nis]
352
353  hostname [-v] [-F filename] [--file filename] [hostname]
354

```

355 `hostname [-v] [-h] [-help] [-V] [--versionname]`

Description

356 **hostname** is used to either ~~set or display or~~, with appropriate privileges, set the current host ~~or domain~~ name of the
 357 system. ~~This~~The host name is used by many of the networking programs-applications to identify the machine. ~~The~~
 358 domain name is also used by NIS/YP.

359 When called without any arguments, the program displays the name of the system as returned by the `gethostname(2)`
 360 function.

361 When called with ~~one~~ a *name* argument ~~or with~~, and the user has appropriate privilege, the ~~—file option, the~~
 362 ~~command sets~~command sets the host name ~~or~~.

363 It is not specified if the NIS/YP hostname displayed will be a fully qualified domain name. ~~Note, that only the~~
 364 ~~super user can change the names.~~

Options

365 ~~-a, —alias~~

366 ~~—displays the alias name of the host (if used).~~

367 ~~-d, —domain~~

368 ~~—displays the name of the DNS domain.~~

369 ~~-F, —file filename~~

370 ~~—reads the host name from the specified file. Comments (lines starting with a #) are ignored.~~

371 ~~-f, —fqdn, —long~~

372 ~~—displays the FQDN (Fully Qualified Domain Name).~~

373 ~~-i, —ip address~~

374 ~~—displays the IP address(es) of the host.~~

375 ~~-s, —short~~

376 ~~—displays the short host name. This is the host name cut at the first dot.~~

377 ~~-v, —verbose~~

378 ~~—tells what's going on.~~

379 ~~-y, —yp, —nis~~

380 ~~—displays the NIS domain name. If~~Applications requiring a parameter is given (or ~~file name~~) then root can also set
 381 a new NIS domain.

LSB-Deprecated Options

382 The behaviors specified in this section are expected to disappear from a future version of the LSB; applications
 383 particular format of hostname should only usecheck the non-LSB-deprecated behaviors.

384 | ~~V, version~~

385 | ~~prints version information on standard output and exits successfully~~take appropriate action.

install

Name

386 `install` — copy files and set attributes

Synopsis

387 **install** [*OPTION*]...*option*...] *SOURCE* *DEST* ~~———— (1st format)~~
 388 **install** [*OPTION*]...*option*...] *SOURCE*... *DIRECTORY* ~~———— (2nd format)~~ *DEST*
 389 **install** [-d [*OPTION*]... *DIRECTORY*... ~~———— (3rd format)~~] --directory] [*option*...] *DIRECTORY*...

Description

390 In the first two formats, copy *SOURCE* to *DEST* or multiple *SOURCE*(*s*) to the existing *DIRECTORY*,
 391 while optionally setting permission modes and owner/group/file ownership. In the third format, ~~create all components~~
 392 ~~of the given~~ each *DIRECTORY*(*ies*) and any missing parent directories shall be created.

Standard Options

393 --backup[=*CONTROLMETHOD*]
 394 makes a backup of each existing destination file. *METHOD* may be one of the following:
 395 • *none* or *off* never make backups.
 396 • *numbered* or *t* make numbered backups. A numbered backup has the form "%s.~%d~", *target_name*,
 397 *version_number*. Each backup shall increment the version number by 1.
 398 • *existing* or *nil* numbered if numbered backups exist, or simple otherwise.
 399 • *simple* or *never* append a suffix to the name. The default suffix is '~', but can be overridden by setting
 400 *SIMPLE_BACKUP_SUFFIX* in the environment, or via the *-S* or --*suffix* option.
 401 If no *METHOD* is specified, the environment variable *VERSION_CONTROL* shall be examined for one of the
 402 above. Unambiguous abbreviations of *METHOD* shall be accepted. If no *METHOD* is specified, or if *METHOD* is
 403 empty, the backup method shall default to *existing*.
 404 If *METHOD* is invalid or ambiguous, **install** shall fail and issue a diagnostic message.

405 -b
 406 is like `--backup`, but does not accept an argument `=existing`.

407 -d, --directory
 408 treats all arguments as directory names; creates all components of the specified directories.

409 -D
 410 creates all leading components of *DEST* except the last, then copies *SOURCE* to *DEST*; useful in the 1st format.

411 -g *GROUP*, --group=*GROUP*

412 if the user has appropriate privilege, sets **group** ownership, instead of process' current group. *GROUP* is either a
 413 name in the user group database, or a positive integer, which shall be used as a group-id.

414 **-m MODE, --mode=MODE**
 415 sets permission mode (specified as in **chmod**), instead of the default `rw-r-xr-x`.

416 **-o OWNER, --owner=OWNER**
 417 if the user has appropriate privilege, sets **ownership** (~~super-user only~~). *OWNER* is either a name in the user login
 418 database, or a positive integer, which shall be used as a user-id.

419 **-p, --preserve-timestamps**
 420 ~~applies~~ copies the access/ and modification times of *SOURCE* files to corresponding destination files.

421 **-s, --strip**
 422 strips symbol tables, only for 1st and 2nd formats.

423 **-S SUFFIX, --suffix=SUFFIX**
 424 ~~overrides the usual~~ equivalent to `--backup=existing`, except if a simple **suffix** is required, use *SUFFIX*.

425 **--verbose**
 426 prints the name of each directory as it is created.

427 **-v, --verbose**
 428 print the name of each file before copying it to `stdout`.

install_initd

Name

429 `install_initd` — install an `init.d` file

Synopsis

430 `/usr/lib/lsb/install_initd` *initd_file*

Description

431 **install_initd** ~~installs an `init.d` file~~ shall install a system initialization file that has been copied to the `/etc/init.d` location
 432 or ~~symlink~~. In such that this file shall be run at the appropriate point during system initialization. The **install_initrd**
 433 command is typically called in the **postinstall** script of a package, the program `/usr/lib/lsb/install_initd` configures a
 434 distribution's boot script system to call the `init.d` file of the package at an appropriate time. See also Section 8.4.

ipcrm

Name

435 | `ipcrm` — ~~provide information on ipc facilities~~ Remove IPC Resources

Synopsis

436 | `ipcrm` ~~[-q msgid | -Q msgkey | -s semid | -S semkey | -m shmid | -M shmkey]...~~
 437 | `ipcrm` [shm | msg | ~~sem~~-msg] id...

Description

438 | ~~`ipcrm` removes the resource~~ (If any of the `-q`, `-Q`, ~~`-s`~~, `-S`, `-m`, or `-M` arguments are given, the `ipcrm` shall behave as
 439 | described in ISO POSIX (2003).
 440 | Otherwise, `ipcrm` shall remove the resource of the `specified` type identified by `id`.

Future Directions

441 | A future revision of this specification may deprecate the second synopsis form.

Rationale

443 | In its first Linux implementation, `ipcrm` used the second syntax shown in the SYNOPSIS. Functionality present in
 444 | other implementations of `ipcrm` has since been added, namely the ability to delete resources by key (not just
 445 | identifier), and to respect the same command line syntax. The previous syntax is still supported for backwards
 446 | compatibility only.

ipcs

Name

447 `ipcs` — provide information on ipc facilities

Synopsis

448 `ipcs` [`[-smq]`] [`[-]`] [`[-tcp]`]

Description

449 `ipcs` provides information on the ipc facilities for which the calling process has read access.

Resource display options

450 `-m`
 451 shared memory segments.
 452 `-q`
 453 message queues.
 454 `-s`
 455 semaphore arrays.

Output format options

456 `-t`
 457 time.
 458 `-p`
 459 pid.
 460 `-c`
 461 creator.

Application Usage

462 In some implementations of `ipcs` the `-a` option will print all information available. In other implementations the `-a`
 463 option will print all resource types. Therefore, applications shall not use the `-a` option.

464 Some implements of `ipcs` implement more output formats than are specified here. These options are not consistent
 465 between differing implementations of `ipcs`. Therefore, only the `-t -c` and `-p` option flags may be used. At least one of
 466 the `-t -c` and `-p` options shall be specified.

killall

Name

467 **killall** — kill processes by name

Synopsis

468 **killall** [-egiqvw] [-signal] name—...
 469 **killall** -l
 470 **killall** -V

Description

471 **killall** sends a signal to all processes running any of the specified commands. If no signal name is specified, **SIGTERM**
 472 is sent.

473 Signals can be specified either by name (e.g. **-HUP**) or by number (e.g. **-1**). Signal 0 (check if a process exists) can
 474 only be specified by number.

475 If the command name contains a slash (/), processes executing that particular file will be selected for killing,
 476 independent of their name.

477 **killall** returns a non-zero return code if no process has been killed for any of the listed commands. If at least one
 478 process has been killed for each command, **killall** returns zero.

479 A **killall** process never kills itself (but may kill other **killall** processes).

Standard Options

480 **-e**

481 requires an exact match for very long names. If a command name is longer than 15 characters, the full name may
 482 be unavailable (i.e. it is swapped out). In this case, **killall** will kill everything that matches within the first 15
 483 characters. With **-e**, such entries are skipped. **killall** prints a message for each skipped entry if **-v** is specified in
 484 addition to **-e**.

485 **-g**

486 kills the process group to which the process belongs. The kill signal is only sent once per group, even if multiple
 487 processes belonging to the same process group were found.

488 **-i**

489 asks interactively for confirmation before killing.

490 **-l**

491 lists all known signal names.

492 **-q**

493 does not complain if no processes were killed.

494 -v
495 reports if the signal was successfully sent.

LSB Deprecated Options

496 The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should
497 only use the non-LSB-deprecated behaviors.

498 -V
499 displays version information.

lpr

Name

500 lpr — off line print

Synopsis

501 **lpr** [-l] [-p] [-Pprinter] [-h] [-s] [-#copies] [-J name]

502 | — [-T title] [name]

Description

503 **lpr** uses a spooling daemon to print the named files when facilities become available. If no names appear, the standard
504 input is assumed.

Standard Options

505 **-l**
506 identifies binary data that is not to be filtered but sent as raw input to printer.

507 **-p**
508 formats with "pr" before sending to printer.

509 **-Pprinter**
510 sends output to the printer named printer instead of the default printer.

511 **-h**
512 suppresses header page.

513 **-s**
514 uses symbolic links.

515 **-#copies**
516 specifies copies as the number of copies to print.

517 **-J name**
518 specifies name as the job name for the header page.

519 **-T title**
520 specifies title as the title used for "pr".

ls

Name

521 `ls` — list directory contents

Description

522 `ls` is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with differences listed below.

Differences

523 `-l`

524 If the file is a character special or block special file, the size of the file shall be replaced with two unsigned
525 numbers in the format "`%u, %u`", representing the major and minor device numbers associated with the special
526 file.

527 The LSB does not specify the meaning of the major and minor devices numbers.

528 `-p`

529 in addition to the ~~Single UNIX Specification~~ ISO POSIX (2003) behavior of printing a slash for a directory, `ls -p`
530 may display other characters for other file types.

531 Certain aspects of ~~internationalized filename globbing~~ the pattern matching notation are optional; see
532 Internationalization and ~~Filename Globbing~~ Pattern Matching Notation➤.

lsb_release

Name

533 `lsb_release` — print distribution specific information

Synopsis

534 `lsb_release` [OPTION]...]

Description

535 The **lsb_release** command prints certain LSB (Linux Standard Base) and Distribution information.

536 If no option, same as are given, the `-v` option is assumed.

Options

537 `-v, --version`

538 displays version of LSB against which distribution is compliant. The version is expressed as a colon seperated list
539 of LSB module descriptions. LSB module descriptions are dash seperated tuples containing the module name,
540 version, and architecture name. The output is a single line of text of the following format:

541 `LSB Version:\t<ListAsDescribedAbove>`

542 `-i, --id`

543 displays string id of distributor. The output is a single line of text of the following format:

544 `Distributor ID:\t<DistributorID>`

545 `-d, --description`

546 displays single line text description of distribution. The output is of the following format:

547 `Description:\t<Description>`

548 `-r, --release`

549 displays release number of distribution. The output is a single line of text of the following format:

550 `Release:\t<Release>`

551 `-c, --codename`

552 displays codename according to distribution release. The output is a single line of text of the following format.

553 `Codename:\t<Codename>`

554 `-a, --all`

555 displays all of the above information.

556 `-s, --short`

557 displays all of the above information in short output format.

558 -h, --help

559 displays a human-readable help message.

Examples

560 The following command will list the LSB Profiles which are currently supported on this platform.

561 example% lsb_release -v

562 | LSB Version: core-2.0-ia32:core-2.0-noarch:graphics-2.0-ia32:graphics-2.0-noarch

m4

Name

563 m4 — macro processor

Description

564 | **m4** is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with extensions as listed below.

Extensions

565 -P

566 | forces a ~~m4_~~ prefix to all builtins to be prefixed with m4_. For example, define becomes m4_define.

567 -I *directory*

568 Add *directory* to the end of the search path for includes.

md5sum

Name

569 | `md5sum` — generates or checks MD5 message digests

Synopsis

570 | `md5sum [-b] [-c [file]] [-t] | file...`

Description

571 | For each file, write to standard output a line containing the MD5 ~~checksum~~ message digest of that file, followed by one
 572 | or more blank characters, followed by the name of the file. The MD5 ~~checksum~~ message digest shall be calculated
 573 | according to ~~RFC1321~~ RFC 1321: The MD5 Message-Digest Algorithm and output as 32 hexadecimal digits (as
 574 | ~~RFC1321 does~~).

575 | If no file names are specified as operands, read from standard input and use "-" as the file name in the output.

Options

576 | `-b`

577 | — ~~uses binary mode.~~

578 | `-c [file]`

579 | checks ~~md5sum~~ the MD5 message digest of all files listed named in *file* against the ~~checksum~~ message digest
 580 | listed in the same file. The actual format of ~~that file~~ is the same as the output of **md5sum**. That is, each line in
 581 | the file describes a file. If *file* is not specified, read message digests from `stdin`.

Exit Status

582 | **md5sum** shall exit with status 0 if the sum was generated successfully, or, in check mode, if the check matched.
 583 | Otherwise, **md5sum** shall exit with a non-zero status.

mknod

Name

584 | **mknod** — make ~~block or character~~ special files

Synopsis

585 | **mknod** [~~OPTION~~]... ~~NAME TYPE~~ [~~MAJOR MINOR~~ *mode* | ~~--mode=mode~~] *name type* [major minor]
 586 | **mknod** [--version]

Description

587 | ~~Create the special file NAME of the given TYPE.~~
 588 | ~~MAJOR MINOR are forbidden for TYPE p, mandatory otherwise. TYPE may be:~~
 589 | The **mknod** command shall create a special file named *name* of the given *type*.
 590 | The *type* shall be one of the following:

591 | **b**
 592 | creates a block (buffered) special file with the specified *major* and *minor* device numbers.
 593 | **c, u**
 594 | creates a character (unbuffered) special file with the specified *major* and *minor* device numbers.
 595 | **p**
 596 | creates a FIFO.

Standard Options

597 | ~~-m, --mode=MODE~~
 598 | ~~—sets permission mode (as in **chmod**), not a=rw—umask.~~

Options

599 | ~~-m mode, --mode=mode~~
 600 | create the special file with file access permissions set as described in *mode*. The permissions may be any absolute
 601 | value (i.e. one not containing '+' or '-') acceptable to the **chmod** command.
 602 | ~~--version~~
 603 | outputs version information and exits.
 604 | This option may be deprecated in a future release of this specification.
 605 | If *type* is **p** parameter, *major* and *minor* shall not be specified. Otherwise, these parameters are mandatory.

Future Directions

This command may be deprecated in a future version of this specification. The *major* and *minor* operands are insufficiently portable to be specified usefully here. Only a FIFO can be portably created by this command, and the **mkfifo** command is a simpler interface for that purpose.

mktemp

Name

mktemp — make temporary file name (unique)

Synopsis

mktemp [-q] [-u] template

Description

The **mktemp** command takes the given file name *template* and overwrites a portion of it to create a file name. This file name is shall be unique and suitable for use by the application.

The *template* should have at least six trailing 'x' characters. These characters are replaced with characters from the portable filename character set in order to generate a unique name.

If **mktemp** can successfully generate a unique file name, and the *-u* option is not present, the file shall be created with read and write permission only for the current user. The **mktemp** command shall write the filename generated to the standard output.

Options

-q

— fails silently if an error occurs. This is useful if a script does not want error output to go to standard error.

fail silently if an error occurs. Diagnostic messages to *stderr* are suppressed, but the command shall still exit with a non-zero exit status if an error occurs.

-u

operates in 'unsafe' mode. The *temp*A unique name is generated, but the temporary file will shall be unlinked before **mktemp** exits. This is slightly better than **mktemp(3)** but still introduces a race condition. Use of this option is not encouraged.

more

Name

626 ~~more — file perusal filter for crt viewing~~
 627 ~~more — display files on a page-by-page basis~~

Description

628 ~~**more** is as specified in the Single UNIX Specification~~ISO POSIX (2003), but with differences as listed below.

Differences

629 The **more** command need not respect the `LINES` and `COLUMNS` environment variables.
 630 ~~The **more** command need not support the following interactive commands:~~

```

$
G
H
control-u
control-f
newline
j
k
r
R
m
'(return to mark)
/
?
N
:e
:t
control-g
ZZ

```

631

632 The following additional options may be supported:

633 `-num`

634 specifies an integer which is the screen size (in lines).

635 `+num`636 starts at line number *num*.637 `+/pattern`

638 Start at the first line matching the pattern, equivalent to executing the search forward (/) command with the given
 639 pattern immediately after opening each file.

640 The following options from ISO POSIX (2003) may behave differently:

641 `-e`

642 has unspecified behavior.

643 `-i`

644 has unspecified behavior.

645 `-n`

646 has unspecified behavior.

647 `-p`

648 Either (1) clear the whole screen and then display the before displaying any text (instead of the usual scrolling
 649 behavior), or (2) provide the behavior specified by the Single UNIX Specification ISO POSIX (2003). In the latter
 650 case, the syntax is "`-p command`".

```

651  -t
652      has unspecified behavior.
653  +num
654  — starts at line number num.
655  +/string
656  — specifies a string that will be searched for before each file is displayed. The more command need not support
657  the following interactive commands:
    g
    G
    u
    control u
    control f
    newline
    j
    k
    r
    R
    m
    ' (return to mark)
    /!
    ?
    N
    :e
    :t
    control g
658  ZZ

```

Rationale

```

659  The +num and +/string options are deprecated in the Single UNIX Specification, Version 2 (SUSv2), and have been
660  removed in ISO POSIX (2003); however we shall continue this specification continues to specify them because the
661  publicly available util-linux-2.11f package does not support the replacement (-p command). The +command
662  option as found in the Single UNIX Specification SUSv2 is more general than what we specify is specified here, but the
663  util-linux-2.11f package appears to only support the more specific +num and +/string forms.

```

mount

Name

664 `mount` — mount a file system

Synopsis

```
665 mount [-hV]
666 mount [-a] [-fFnrsvw] [-t vfstype]
667 mount [-fnrsvw] [-o options [...]] [device | dir]
668 mount [-fnrsvw] [-t vfstype] [-o options] device dir
```

Description

669 Files—As described in ISO POSIX (2003), all files in the system are named organized in a big tree, directed graph,
 670 known as the file hierarchy, rooted at /. These files can be spread out over several underlying devices. The **mount**
 671 serves to command shall attach the file system found on some underlying device to the big file tree. Conversely,
 672 ~~umount(8) will detach it again.~~ hierarchy.

Standard Options

```
673 -v
674 | invokes verbose mode. The mount command shall provide diagnostic messages on stdout.

675 -a
676 | mounts all filesystems (of the given types) mentioned in /etc/fstab.

677 -F
678 | combines with If the -a -a option is also present, fork off a new incarnation of mount for each device to be
679 | mounted. This will do the mounts on different devices or different NFS servers in parallel.

680 -f
681 | causes everything to be done except for the actual system call; if it's not obvious, this `fakes' mounting the file
682 | system.

683 -n
684 | mounts without writing in /etc/mtab. This is necessary for example when /etc is on a read-only file system.

685 -s
686 | tolerates sloppy mount options rather than failing. This will ignore mount options not supported by a filesystem
687 | type. Not all filesystems support this option.

688 -r
689 | mounts the file system read-only. A synonym is -o ro.
```

```

690 -w
691 | mounts the file system read/write. (default) A synonym is -o rw.
692 -L label
693 | mountsIf the file /proc/partitions is supported, mount the partition that has the specified label.
694 -U uuid
695 | mountsIf the file /proc/partitions is supported, mount the partition that has the specified uuid. These two
696 | options require the file /proc/partitions to exist.
697 -t vfstype
698 | indicates a file system type of vfstype.
699 More than one type may be specified in a comma separated list. The list of file system types can be prefixed with
700 no to specify the file system types on which no action should be taken.
701 -o
702 options are specified with a -o flag followed by a comma-separated string of options. Some of these options are
703 only useful when they appear in the /etc/fstab file. The following options apply to any file system that is
704 being mounted:
705 async
706 | doesperform all I/O to the file system asynchronously.
707 atime
708 | updates inode access time for each access. (default)
709 auto
710 | in /etc/fstab, indicate the device is mountable with -a.
711 defaults
712 | uses default options: rw, suid, dev, exec, auto, nouser, and -async.
713 dev
714 | interprets character or block special devices on the file system.
715 exec
716 | permits execution of binaries.
717 noatime
718 | does not update inode file access times on this file system.
719 noauto
720 | in /etc/fstab, indicates the device is only explicitly mountable.
721 nodev

```


722		does not interpret character or block special devices on the file system.
723		noexec
724		does not allow execution of any binaries on the mounted file system.
725		nosuid
726		does not allow set-user-identifier or set-group-identifier bits to take effect.
727		nouser
728		forbids an ordinary (i.e., non-root) unprivileged user to mount the file system. (default)
729		remount
730		attempts to remount an already-mounted file system. This is commonly used to change the mount
731		flags options for a file system, especially to make a read-only file system writable.
732		ro
733		mounts the file system read-only.
734		rw
735		mounts the file system read-write.
736		suid
737		allows set-user-identifier or set-group-identifier bits to take effect.
738		sync
739		does all I/O to the file system synchronously.
740		user
741		allows an ordinary unprivileged user to mount the file system. This option implies the options noexec ,
742		nosuid , and nodev (unless overridden by subsequent options, as in the option line user,exec,dev,suid).

LSB Deprecated Options

743		The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should
744		only use the non-LSB-deprecated behaviors.
745		-V
746		outputs version and exit.

msgfmt

Name

747 msgfmt — create a message object from a message file

Synopsis

748 **msgfmt** [-options...] *filename.po*...

Description

749 The **msgfmt** command generates a binary message object file catalog from a textual translation
750 description. Message catalogs, or message object files, are stored in files with a .mo extension.

751 The format of message object files (*filename.po*), without changing is not guaranteed to be portable. Message
752 catalogs should always be generated on the target architecture using the **msgfmt** command.

753 The source message files, otherwise known as portable object files, have a .po extension.

754 The *filename* operands shall be portable object files. The .po file contains messages to be displayed to users by
755 system utilities or by application programs. ~~po files can be edited~~ The portable object files are text files, and the
756 messages in them can be rewritten in any language supported by the system.

757 If ~~input~~ any *filename* is -, a portable object file is -, shall be read from the standard input ~~is read~~.

758 The ~~xgettext utility can be used to create po files from script or programs.~~

759 **msgfmt** command interprets data as characters according to the current setting of the LC_CTYPE locale category.

Options

760 -c

761 --check

762 Detect and diagnose input file anomalies which might represent translation errors. The msgid and msgstr
763 strings are studied and compared. It is considered abnormal that one string starts or ends with a newline while the
764 other does not.

765 If the message is flagged as c-format (see Comment Handling), check that the msgid string and the msgstr
766 translation have the same number of % format specifiers, with matching types.

767 -D *directory*

768 --directory=*directory*

769 Add *directory* to list for input files search. If *filename* is not an absolute pathname and *filename* cannot be
770 opened, search for it in *directory*. This option may be repeated. Directories shall be searched in order, with
771 the leftmost *directory* searched first.

772 -f

773 --use-fuzzy

774 Use ~~fuzzy~~ entries marked as `fuzzy` in output. If this option is not specified, ~~fuzzy~~ such entries are not included
 775 into the output. See Comment Handling below.

776 `-o output-file`
 777 `--output-file=output-file`

778 Specify the output file name as `output-file`. If multiple domains or duplicate msgids in the `.po` file are present,
 779 the behavior is unspecified. If `output-file` is `-`, output is written to standard output.

780 `-S`
 781 `--strict`

782 — Direct the utility to work strictly following the UniForum/Sun implementation. Currently this only affects the
 783 naming of the output file. If this option is not given the name of the output file is the same as the domain name. If
 784 the strict UniForum mode is enabled the suffix `.mo` is added to the file name if it is not already present.

785 Ensure that all output files have a `.mo` extension. Output files are named either by the `-o` (or `--output-file`)
 786 option, or by domains found in the input files.

787 `-v`
 788 `--verbose`

789 — Detect and diagnose input file anomalies which might represent translation errors. The `msgid` and `msgstr` strings
 790 are studied and compared. It is considered abnormal that one string starts or ends with a newline while the other
 791 does not.

792 Also, if the string represents a format string used in a `printf` like function both strings should have the same
 793 number of `%` format specifiers, with matching types. If the flag `c` format or possible `e` format appears in the
 794 special comment `#`, for this entry a check is performed. For example, the check will diagnose using `%. *s`
 795 against `%s`, or `%d` against `%s`, or `%d` against `%x`. It can even handle positional parameters.

796 Print additional information to the standard error, including the number of translated strings processed.

Operands

797 The `filename.po` operands are treated as portable object files. The format of portable object files is defined in
 798 EXTENDED DESCRIPTION.

Standard Input

799 The standard input is not used unless a `filename.po` operand is specified as `"-"`.

Environment Variables

800 LANGUAGE

801 Specifies one or more locale names. ~~See `gettext` message handling functions for more information.~~

802 LANG

803 Specifies locale name.

804 LC_ALL

805 Specifies locale name for all categories. If defined, overrides LANG, LC_CTYPE and LC_MESSAGES.

806 **LC_CTYPE**
 807 — Specifies locale name for character handling.
 808 Determine the locale for the interpretation of sequences of bytes of text data as characters (for example,
 809 single-byte as opposed to multi-byte characters in arguments and input files).

810 **LC_MESSAGES**
 811 Specifies messaging locale, and if present overrides LANG for messages.

Standard Output

812 The standard output is not used unless the option-argument of the `-o` option is specified as `-`.

Extended Description

813 The format of portable object files (`.po` files) is defined as follows. Each `.po` file contains one or more lines, with
 814 each line containing either a comment or a statement. Comments start the line with a hash mark (`#`) and end with the
 815 newline character. ~~All comments and empty lines are,~~ or lines containing only white-space, shall be ignored.
 816 Comments can in certain circumstances alter the behavior of **msgfmt**. See Comment Handling below for details on
 817 comment processing. The format of a statement is:

818 directive value

819 Each directive starts at the beginning of the line and is separated from value by white space (such as one or more
 820 space or tab characters). The value consists of one or more quoted strings separated by white space. If two or more
 821 strings are specified as value, they are normalized into single string using the string normalization syntax ~~the same as~~
 822 ~~the ISO C language. Use any of the~~ specified in ISO C (1999). The following ~~types of~~ directives are supported:

823 domain domainname

824 msgid message_identifier

825 msgid_plural untranslated_string_plural

826 msgstr message_string

827 msgstr[n] message_string

828 The behavior of the domain directive is affected by the options used. See OPTIONS for the behavior when the `-o`
 829 option is specified. If the `-o` option is not specified, the behavior of the domain directive is as follows: ↵

830 1)↵. All msgids from the beginning of each `.po` file to the first domain directive are put into a default message object
 831 file, messages (or messages.mo if the `--strict` option is specified).↵

832 2)↵. When **msgfmt** encounters a domain domainname directive in the `.po` file, all following *msgids* until the next
 833 domain directive are put into the message object file domainname (or domainname.mo if `--strict` option is
 834 specified).↵

835 3)↵. Duplicate *msgids* are defined in the scope of each domain. That is, a *msgid* is considered a duplicate only if the
 836 identical *msgid* exists in the same domain.↵

837 4)↵. All duplicate *msgids* are ignored.

The `msgid` directive specifies the value of a message identifier associated with the directive that follows it. The `msgid_plural` directive specifies the plural form message specified to the plural message handling functions `ngettext()`, `dngettext()` or `dcngettext()`. The `message_identifier` string identifies a target string to be used at retrieval time. Each statement containing a `msgid` directive shall be followed by a statement containing a `msgstr` directive or `msgstr[n]` directives.

The `msgstr` directive specifies the target string associated with the `message_identifier` string declared in the immediately preceding `msgid` directive.

The `msgstr[n]` (where $n = 0, 1, 2, \dots$) directive specifies the target string to be used with plural form handling functions `ngettext()`, `dngettext()` and `dcngettext()`.

Message strings can contain the following escape sequences: `\n` for newline, `\t` for tab, `\v` for vertical tab, `\b` for backspace, `\r` for carriage return, `\f` for formfeed, `\\` for backslash, `\"` for double quote, `\ddd` for octal bit pattern:

Table 3-1. Escape Sequences

<code>\n</code>	newline
<code>\t</code>	tab
<code>\v</code>	vertical tab
<code>\b</code>	backspace
<code>\r</code>	carriage return
<code>\f</code>	formfeed
<code>\\</code>	backslash
<code>\"</code>	double quote
<code>\ddd</code>	octal bit pattern
<code>\xHH</code>	hexadecimal bit pattern

Comment Handling

Comments are introduced by a `#`, and `\xHH` for hexadecimal bit pattern.

Comments should be in one line. The second character (i.e. the character following the `#`) has special meaning. Regular comments should follow a space character. Other comment types include:

`#translator` normal-comments

`#.` automatic-comments

`#:` reference...

`#,` flag

The comments that starts with `#.` and `#:` are automatically generated by `xgettext` utility. The `#:` comments indicate the location of the `msgid` string in the source files in `filename:line` format. The `#.` comments are generated when `-c` option of the `xgettext` utility is specified. These comments are informative only and silently ignored by the `msgfmt` utility.

The `#,` comments requires one or more flags separated by comma (,) character. The following flags can be specified:

Automatic and reference comments are typically generated by external utilities, and are not specified by the LSB. The **msgfmt** command shall ignore such comments.

Portable object files may be produced by unspecified tools. Some of the comment types described here may arise from the use of such tools. It is beyond the scope of this specification to describe these tools.

The #, comments require one or more flags separated by the comma (,) character. The following flags can be specified:

fuzzy

~~This flag can be generated by the msgmerge utility or can be inserted by the translator. It shows that the following msgstr string might not be a correct translation (anymore). Only the translator (i.e. the individual undertaking the translation) can judge if the translation requires further modification, or is acceptable as is. Once satisfied with the translation, the translator then removes this fuzzy flag. The msgmerge programs inserts this when it combined the msgid and msgstr entries after fuzzy search only.~~

If this flag is specified, the **msgfmt** utility will not generate the entry for the immediately following msgid in the output message catalog, unless the `--use-fuzzy` is specified.

c-format

no-c-format

~~The flags are automatically added by the xgettext utility and they should not be added manually. The c-format flag indicates that the msgid string is used as format string by printf-like functions. In case the c-format flag is given for a string the msgfmt utility does some more may perform additional tests to check to validity of the translation.~~

Plurals

The msgid entry with empty string ("") is called the header entry and is treated specially. If the message string for the header entry contains `nplurals=value`, the value indicates the number of plural forms. For example, if `nplurals=4`, there are 4 plural forms. If `nplurals` is defined, there should be a `plural=expression` ~~in~~ on the same line, separated by a semicolon (;) character. The expression is a C language expression to determine which version of `msgstr[n]` to be used based on the value of `n`, the last argument of `ngettext()`, `dngettext()` or `dcngettext()`. For example:

```
nplurals=2; plural=n == 1 ? 0 : 1
```

indicates that there are 2 plural forms in the language; `msgstr[0]` is used if `n == 1`, otherwise `msgstr[1]` is used. Another example:

```
nplurals=3; plural=n==1 ? 0 : n==2 ? 1 : 2
```

indicates that there are 3 plural forms in the language; `msgstr[0]` is used if `n == 1`, `msgstr[1]` is used if `n == 2`, otherwise `msgstr[2]` is used.

If the header entry contains `charset=codeset` string, the `codeset` is used to indicate the codeset to be used to encode the message strings. If the output string's codeset is different from the message string's codeset, codeset conversion from the message strings's codeset to the output string's codeset will be performed upon the call of `gettext()`, `dgettext()`, `dcgettext()`, `ngettext()`, `dngettext()`, and `dcngettext()`. The output string's codeset is determined by the current locale's codeset (the return value of `nl_langinfo(CODESET)`) by default, and can be changed by the call of `bind_textdomain_codeset()`.

Exit Status

902 The following exit values are returned:

903 0

904 Successful completion.

905 >0

906 An error occurred.

Application Usage

907 Neither **msgfmt** nor any `gettext()` routine function imposes a limit on the total length of a message. Installing
 908 message catalogs under the C locale is pointless, since they are ignored for the sake of efficiency.

Examples

909 Example 1: Examples of creating message objects from message files.

910 In this example `module1.po` and `module2.po` and `module3.po` are portable message objects files.

```

911 example% cat module1.po
912
913 # default domain "messages"
914
915 msgid "msg-1message one"
916
917 msgstr "msg-1 translationmensaje número uno"
918
919 #
920
921 domain "help_domain"
922
923 msgid "help 2two"
924
925 msgstr "help-2 translationayuda número dos"
926
927 #
928
929 domain "error_domain"
930
931 msgid "error 3three"
932
933 msgstr "error 3 translationnúmero tres"
934
935
936 example% cat module2.po
937
938 # default domain "messages"
939
940 msgid "msg-4message four"
```

```
941 | msgstr "msg 4 translationmensaje número cuatro"
```

```
942
```

```
943 | #
```

```
944
```

```
945 | domain "error_domain"
```

```
946
```

```
947 | msgid "error 5five"
```

```
948
```

```
949 | msgstr "error 5 translationnúmero cinco"
```

```
950
```

```
951 | #
```

```
952
```

```
953 | domain "window_domain"
```

```
954
```

```
955 | msgid "window 6six"
```

```
956
```

```
957 | msgstr "ventana número seises"
```

```
958 | example% cat module3.po
```

```
959
```

```
960 | # default domain "messages"
```

```
961
```

```
962 | msgid "message seven"
```

```
963
```

```
964 | msgstr "mensaje número siete"
```

```
965
```

```
966 | msgstr "window 6 translation"
```

```
967 | The following command will produce the output files; messages, help_domain, and error_domain.
```

```
968 | example% msgfmt module1.po
```

```
969 | The following command will produce the output files; messages, help_domain, error_domain, and
```

```
970 | window_domain.
```

```
971 | example% msgfmt module1.po module2.po
```

```
972 | The following example will produce the output file hello.mo.
```

```
973 | example% msgfmt -o hello.mo module1.po module2module3.po
```


newgrp

Name

974 newgrp — change group ID

Synopsis

975 **newgrp** [-] [group]

Description

976 ~~newgrp changes the current group ID during a login session. If the optional - flag is given, the user's environment will~~
977 ~~be reinitialized as though the user had logged in, otherwise the current environment, including current working~~
978 ~~directory, remains unchanged.~~

979 The **newgrp** command is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

981 The `-l` option specified in ISO POSIX (2003) need not be supported.

od

Name

982 **od** — dump files in octal and other formats

Synopsis

983 **od** [-abcdfilox] [-w *width* | --width=*width*] [-v] [-A *address_base*] [-j *skip*] [-n *count*] [-t *type_string*]
 984 [file...]
 985 **od** --traditional [options] [file] [[+]offset [.] [b]] [[+]label [.] [b]]

Description

986 **od** is as specified in the Single UNIX Specification ISO POSIX (2003), but with extensions and differences as listed below.

Extensions

~~-w~~ Differences

987 ~~-w~~*width*, --width[=~~BYTES~~]*width*]

988 ~~outputs BYTES bytes per~~each output line is limited to *width* bytes from the input.

989 --traditional

990 accepts arguments in ~~pre-POSIX~~traditional form.

991 The XSI optional behavior described in ISO POSIX (2003) is not supported unless the *--traditional*
 992 option is also specified.

Pre-POSIX and XSI Specifications

993 The LSB supports option intermixtures with the following pre-POSIX specifications and XSI options:

994 -a

995 is equivalent to *-t a*, selects named characters.

996 -b

997 is equivalent to *-t o1*, selects octal bytes.

998 -c

999 is equivalent to *-t c*, selects characters.

1000 -d

1001 is equivalent to *-t u2*, selects unsigned decimal two byte units.

1002 -f

1003 is equivalent to `-t fF`, selects floats.

1004 ~~`-h`~~

1005 ~~is equivalent to `-t x2`, selects hexadecimal shorts.~~

1006 `-i`

1007 is equivalent to `-t d2`, selects decimal shorts two byte units.

1008 This usage may change in future releases; portable applications should use `-t d2`.

1009 `-l`

1010 is equivalent to `-t d4`, selects decimal longs.

1011 `-o`

1012 is equivalent to `-t o2`, selects octal two byte units.

1013 `-x`

1014 is equivalent to `-t x2`, selects hexadecimal two byte units.

1015 Note that the XSI option `-s` need not be supported.

Traditional Usage

1016 If the `--traditional` is specified, there may be between zero and three operands specified.

1017 If no operands are specified, then `od` shall read the standard input.

1018 If there is exactly one operand, and it is an offset of the form `[+]offset[.][b]`, then it shall be interpreted as

1019 specified in ISO POSIX (2003). The file to be dumped shall be the standard input.

1020 If there are exactly two operands, and they are both of the form `[+]offset[.][b]`, then the first shall be an treated as

1021 an offset (as above), and the second shall be a label, in the same format as the offset. If a label is specified, then the first

1022 output line produced for each input block shall be preceded by the input offset, cumulative across input files, of the

1023 next byte to be written, followed by the label, in parentheses. The label shall increment in the same manner as the

1024 offset.

1025 If there are three operands, then the first shall be the file to dump, the second the offset, and the third the label.

passwd

Name

1026 `passwd` — change user password

Synopsis

1027 **passwd** [-x max] [-n min] [-w warn] [-i inact] name
 1028 **passwd** {-l+ | -u} name

Description

1029 **passwd** changes passwords for user and group accounts. A normal user may only change the password for their own
 1030 account, the super user may change the password for any account. **passwd** also changes password expiry dates and
 1031 intervals. Applications may not assume the format of prompts and anticipated input for user interaction, because they
 1032 are unspecified.

Options

1033 **-x max**
 1034 sets the maximum number of days a password remains valid.

1035 **-n min**
 1036 sets the minimum number of days before a password may be changed.

1037 **-w warn**
 1038 sets the number of days warning the user will receive before their password will expire.

1039 **-i inactive**
 1040 disables an account after the password has been expired for the given number of days.

1041 **-l**
 1042 disables an account by changing the password to a value which matches no possible encrypted value.

1043 **-u**
 1044 re-enables an account by changing the password back to its previous value.

patch

Name

1045 `patch` — apply a diff file to an original

Description

1046 **patch** is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with extensions as listed below.

Extensions

1047 `--binary`

1048 reads and write all files in binary mode, except for standard output and /dev/tty. This option has no effect on
1049 POSIX-compliant systems.

1050 `-u, --unified`

1051 interprets the patch file as a unified context diff.

pidof

Name

1052 `pidof` — find the process ID of a running program

Synopsis

1053 **pidof** [-s] [-x] [-o omitpid...] program ~~[program...]~~...

Description

1054 Return the process ID of a process which is running the program named on the command line.⁺

Options

1055 `-s`

1056 instructs the program to only return one pid.

1057 `-x`

1058 causes the program to also return process id's of shells running the named scripts.

1059 `-o`

1060 omits processes with specified process id.

Notes

1. ~~Need further investigation on the behavior of various implementations concerning whether program is a full pathname, the basename only, the program as named by argv[0], or what.~~

remove_initd**Name**

`remove_initd` — clean up boot script system modifications introduced by `install_initd`

Synopsis

`/usr/lib/lsb/remove_initd` `initd_file`

Description

remove_initd processes the removal of the modifications made to a distribution's boot script system by the **install_initd** program. This cleanup is performed in the preuninstall script of a package; however, the package manager is still responsible for removing the `/etc/init.d` file. See also Section 8.4.

renice**Name**

`renice` — alter priority of running processes

Description

renice is as specified in ~~the Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

Differences

- `-n` increment
- has unspecified behavior.

sed

Name

1073 sed — stream editor

Description

1074 | **sed** is as specified in the ~~Single UNIX Specification~~ ISO POSIX (2003), but with differences as listed below.

LSB Differences

1075 Certain aspects of internationalized regular expressions are optional; see Internationalization and Regular
1076 Expressions>.

sendmail

Name

1077 `sendmail` — an electronic mail transport agent

Synopsis

1078 `sendmail` [~~flags~~options] [address—...]

Description

1079 To deliver electronic mail (email), applications shall support the interface provided by `/usr/sbin/sendmail` (described
1080 here). This interface shall be the default delivery method for applications.

1081 This program sends an email message to one or more recipients, routing the message as necessary. This program is not
1082 intended as a user interface routine.

1083 With no ~~flags~~options, **sendmail** reads its standard input up to an end-of-file or a line consisting only of a single dot and
1084 sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the
1085 syntax and contents of the addresses.

1086 It is recommended that applications use as few ~~flags~~options as necessary, none if possible.

1087 Some agents allow aliasing on the local system to be prevented by preceding the address with a backslash.

1088 The format of messages shall be as defined in RFC 2822.

Options

1089 `-bm`

1090 reads mail from standard input and delivers to the recipient addresses. This is the default mode of operation.

1091 `-bp`

1092 lists information about messages currently in the input mail queue.

1093 `-bs`

1094 uses the SMTP protocol as described in RFC 2821; reads SMTP commands on standard input and writes SMTP
1095 responses on standard output.

1096 Note that RFC 2821 specifies `\r\n` (CR-LF) be used at the end of each line, but pipes almost always use `\n` (LF)
1097 instead. To deal with this, agents will accept both `\r\n` and `\n` at the end of each line. When accepting `\r\n`, the `\r`
1098 before the `\n` is silently discarded.

1099 `-F fullname`

1100 explicitly sets the full name of the sender for incoming mail unless the message already contains a `From:` message
1101 header.

1102 If the user running **sendmail** is not sufficiently trusted, then the actual sender may be indicated in the message,
1103 depending on the behavior of the agent.

1104 **-f name**
 1105 explicitly sets the envelope sender address for incoming mail. If there is no From: header, the address specified in
 1106 the From: header will also be set.
 1107 If the user running **sendmail** is not sufficiently trusted, then the actual sender will be indicated in the message.

1108 **-i**
 1109 ignores dots alone on lines by themselves in incoming messages. If **-bs** is also used, the behavior is unspecified.

1110 **-odb**
 1111 delivers any mail in background, if supported; otherwise ignored.

1112 **-odf**
 1113 delivers any mail in foreground, if supported; otherwise ignored.

1114 **-oem or -em**
 1115 mails errors back to the sender. (default)

1116 **-oep or -ep**
 1117 writes errors to the standard error output.

1118 **-oeq or -eq**
 1119 does not send notification of errors to the sender. This only works for mail delivered locally.

1120 **-oi**
 1121 is equivalent to **-i**.

1122 **-om**
 1123 indicates that the sender of a message should receive a copy of the message if the sender appears in an alias
 1124 expansion. Ignored if aliases are not supported.

1125 **-t**
 1126 reads the message to obtain recipients from the To:, Cc:, and Bcc: headers in the message instead of from the
 1127 command arguments. If a Bcc: header is present, it is removed from the message unless there is no To: or Cc:
 1128 header, in which case a Bcc: header with no data is created, in accordance with RFC 2822.
 1129 If there are any arguments, they specify addresses to which the message is not to be delivered. That is, the
 1130 argument addresses are removed from the recipients list obtained from the headers. Note: some agents implement
 1131 this behavior in reverse, adding addresses instead of removing them. Others may disallow addresses in argument
 1132 list. Therefore, applications should not put addresses in the argument list if **-t** is used.
 1133 This option is sometimes ignored when not in **-bm** mode (the default).

Exit status

1134 **0**
 1135 successful completion on all addresses. This does not indicate successful delivery.

1136 >0
1137 there was an error.

Notes/Rationale

1138 This page is believed to reflect functionality provided by smail, exim and other implementations, not just the **sendmail**
1139 implementation.

shutdown

Name

1140 `shutdown` — bring the system down

Synopsis

1141 `/sbin/shutdown` [-t sec] [-arkhcfF] time [warning-message]

Description

1142 **shutdown** brings the system down in a secure way. All logged-in users are notified that the system is going down, and
 1143 login(1) is blocked. It is possible to shut the system down immediately or after a specified delay. All processes are first
 1144 notified that the system is going down by the signal SIGTERM. If neither the -h or the -r argument is used, then the
 1145 default behavior is to take the system to runlevel one where administrative tasks can be run.

Standard Options

1146 -a

1147 uses /etc/shutdown.allow.

1148 -t sec

1149 tells init(8) to wait sec seconds between sending processes the warning and the kill signal, before changing to
 1150 another runlevel.

1151 -k

1152 doesn't really shutdown; only sends the warning messages to everybody.

1153 -r

1154 reboots after shutdown.

1155 -h

1156 halts after shutdown. Powering off after halting is unspecified.

1157 -f

1158 skips fsck on reboot.

1159 -F

1160 forces fsck on reboot.

1161 -c

1162 cancels an already running **shutdown**. With this option, it is of course not possible to give the time argument, but
 1163 you can enter a explanatory message on the command line that will be sent to all users.

1164 time

1165 specifies when to shut down.

1166 The time argument can have different formats. First, it can be an absolute time in the format hh:mm, in which hh
1167 is the hour (1 or 2 digits) and mm is the minute of the hour (in two digits). Second, it can be in the format +m, in
1168 which m is the number of minutes to wait. The word now is an alias for +0.

1169 If **shutdown** is called with a delay, it creates the advisory file /etc/nologin which causes programs such as
1170 login(1) to not allow new user logins. **shutdown** only removes this file if it is stopped before it can signal init (i.e.
1171 it is cancelled or something goes wrong). Otherwise it is the responsibility of the system shutdown or startup
1172 scripts to remove this file so that users can login.

1173 warning-message

1174 specifies message to send all users.

su

Name

1175 su — change user ID or become super-user

Synopsis

1176 | **su** [~~OPTION~~Options] [-] [username [ARGS]]

Description

1177 **su** is used to become another user during a login session. Invoked without a username, **su** defaults to becoming the
 1178 super user. The optional argument - may be used to provide an environment similar to what the user would expect had
 1179 the user logged in directly.

1180 The user will be prompted for a password, if appropriate. Invalid passwords will produce an error message. All
 1181 attempts, both valid and invalid, are logged to detect abuses of the system. Applications may not assume the format of
 1182 prompts and anticipated input for user interaction, because they are unspecified.

1183 An optional command can be executed. This is done by the shell specified in /etc/passwd for the target user unless the
 1184 -s or -m options are used. Any arguments supplied after the username will be passed to the invoked shell (shell shall
 1185 support the -c command line option in order for a command to be passed to it).

1186 The current environment is passed to the new shell. The value of \$PATH is reset to /bin:/usr/bin for normal users, or
 1187 /sbin:/bin:/usr/sbin:/usr/bin for the super user. This may be changed with the ENV_PATH and ENV_SUPATH
 1188 definitions in /etc/login.defs. When using the -m or -p options, the user's environment is not changed.

1189 A subsystem login is indicated by the presense of a "*" as the first character of the login shell. The given home
 1190 directory will be used as the root of a new filesystem which the user is actually logged into.

Standard Options

1191 -

1192 makes this a login shell.

1193 -c, --comand=command

1194 passes command to the invoked shell. It is passed directly to the invoked shell (using the shell's -c option), so its
 1195 syntax is whatever that shell can accept.

1196 -m, -p, --preserve-environment

1197 does not reset environment variables, and keeps the same shell if it is present in /etc/shells.

1198 -s, --shell=shell

1199 uses shell instead of the default in /etc/passwd. The shell specified shall be present in /etc/shells.

sync

Name

1200 `sync` — flush filesystem buffers

Synopsis

1201 `sync`

Description

1202 Force changed blocks to disk, update the super block.

tar

Name

1203 `tar` — file archiver

Description

1204 `tar` is as specified in the ~~Single UNIX Specification, Version 2~~SUSv2, but with differences as listed below.

Differences

1205 Certain aspects of internationalized filename globbing are optional; see Internationalization and ~~Filename~~
1206 ~~Globber~~Pattern Matching Notation>.

1207 `-h`

1208 doesn't dump symlinks; dumps the files they point to.

1209 `-Z`

1210 filters the archive through **gzip**.

umount

Name

1211 `umount` — unmount file systems

Synopsis

1212 **umount** [-hV]
 1213 **umount** -a [-nrV] [-t vfstype]
 1214 | **umount** [-nrV] device | dir [+...]

Description

1215 **umount** detaches the file system(s) mentioned from the file hierarchy. A file system is specified by giving the
 1216 directory where it has been mounted.

Standard Options

1217 **-v**
 1218 invokes verbose mode.

1219 **-n**
 1220 unmounts without writing in /etc/mtab.

1221 **-r**
 1222 tries to remount read-only if unmounting fails.

1223 **-a**
 1224 unmounts all of the file systems described in /etc/mtab except for the proc filesystem.

1225 **-t vfstype**
 1226 indicates that the actions should only be taken on file systems of the specified type. More than one type may be
 1227 specified in a comma separated list. The list of file system types can be prefixed with no to specify the file system
 1228 types on which no action should be taken.

1229 **-f**
 1230 forces unmount (in case of an unreachable NFS system).

LSB Deprecated Options

1231 The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should
 1232 only use the non-LSB-deprecated behaviors.

1233 **-V**
 1234 print version and exits.

useradd

Name

1235 useradd — create a new user or update default new user information

Synopsis

```
1236 useradd [-c comment] [-d home_dir]
1237         [-g initial_group] [-G group[,...]]
1238         [-m [-k skeleton_dir]] [-p passwd] [-r]
1239         [-s shell] [-u uid [-o]] login
1240
1241 useradd -D [-g default_group] [-b default_home]
```


1242 [-s default_shell]

Description

1243 When invoked without the -D option, **useradd** creates a new user account using the values specified on the command
 1244 line and the default values from the system. The new user account will be entered into the system files as needed, the
 1245 home directory will be created, and initial files copied, depending on the command line options.

1246 When invoked with the -D option, **useradd** will either display the current default values, or update the default values
 1247 from the command line. If no options are specified, **useradd** displays the current default values.

Standard Options

1248 -c comment

1249 specifies the new user's password file comment field value.

1250 -d home_dir

1251 creates the new user using home_dir as the value for the user's login directory. The default is to append the login
 1252 name to default_home and use that as the login directory name.

1253 -g initial_group

1254 specifies the group name or number of the user's initial login group. The group name shall exist. A group number
 1255 shall refer to an already existing group. If -g is not specified, the implementation will follow the normal user
 1256 default for that system. This may create a new group or choose a default group that normal users are placed in.
 1257 Applications which require control of the groups into which a user is placed should specify -g.

1258 -G group,[...]

1259 specifies a list of supplementary groups which the user is also a member of. Each group is separated from the next
 1260 by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given
 1261 with the -g option. The default is for the user to belong only to the initial group.

1262 -m [-k skeleton_dir]

1263 specifies the user's home directory will be created if it does not exist. The files contained in skeleton_dir will be
 1264 copied to the home directory if the -k option is used, otherwise the files contained in /etc/skel will be used instead.
 1265 Any directories contained in skeleton_dir or /etc/skel will be created in the user's home directory as well. The -k
 1266 option is only valid in conjunction with the -m option. The default is to not create the directory and to not copy
 1267 any files.

1268 -p passwd

1269 is the encrypted password, as returned by crypt(3). The default is to disable the account.

1270 -r

1271 creates a system account, that is, a user with a UID in the range reserved for system account users. If there is not
 1272 a UID free in the reserved range the command will fail.

1273 -s shell

1274 specifies the name of the user's login shell. The default is to leave this field blank, which causes the system to
1275 select the default login shell.

1276 -u uid [-o]

1277 specifies the numerical value of the user's ID. This value shall be unique, unless the -o option is used. The value
1278 shall be non-negative. The default is the smallest ID value greater than 499 which is not yet used.

Change Default Options

1279 -b default_home

1280 specifies the initial path prefix for a new user's home directory. The user's name will be affixed to the end of
1281 default_home to create the new directory name if the -d option is not used when creating a new account.

1282 -g default_group

1283 specifies the group name or ID for a new user's initial group. The named group shall exist, and a numerical group
1284 ID shall have an existing entry.

1285 -s default_shell

1286 specifies the name of the new user's login shell. The named program will be used for all future new user accounts.

1287 -c comment

1288 specifies the new user's password file comment field value.

Application Usage

1289 The -D option will typically be used by system administration packages. Most applications should not change defaults
1290 which will affect other applications and users.

userdel

Name

1291 `userdel` — delete a user account and related files

Synopsis

1292 `userdel [-r] login`

Description

1293 Delete the user account named *login*. If there is also a group named *login*, this command may delete the group as
 1294 well, or may leave it alone.

Options

1295 `-r`
 1296 removes files in the user's home directory along with the home directory itself. Files located in other file system
 1297 will have to be searched for and deleted manually.

usermod

Name

1298 `usermod` — modify a user account

Synopsis

1299 `usermod [-c comment] [-d home_dir [-m]]`
 1300 `[-g initial_group] [-G group[,...]]`
 1301 `[-l login_name] [-p passwd]`

1302 [-s shell] [-u uid [-o]] login

Options

1303 -c comment

1304 specifies the new value of the user's password file comment field.

1305 -d home_dir

1306 specifies the user's new login directory. If the -m option is given the contents of the current home directory will be
1307 moved to the new home directory, which is created if it does not already exist.

1308 -g initial_group

1309 specifies the group name or number of the user's new initial login group. The group name shall exist. A group
1310 number shall refer to an already existing group.

1311 -G group,[...]

1312 specifies a list of supplementary groups which the user is also a member of. Each group is separated from the next
1313 by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given
1314 with the -g option. If the user is currently a member of a group which is not listed, the user will be removed from
1315 the group.

1316 -l login_name

1317 changes the name of the user from login to login_name. Nothing else is changed. In particular, the user's home
1318 directory name should probably be changed to reflect the new login name.

1319 -p passwd

1320 is the encrypted password, as returned by crypt(3).

1321 -s shell

1322 specifies the name of the user's new login shell. Setting this field to blank causes the system to select the default
1323 login shell.

1324 -u uid [-o]

1325 specifies the numerical value of the user's ID. This value shall be unique, unless the -o option is used. The value
1326 shall be non-negative. Any files which the user owns and which are located in the directory tree rooted at the
1327 user's home directory will have the file user ID changed automatically. Files outside of the user's home directory
1328 shall be altered manually.

xargs

Name

1329 `xargs` — build and execute command lines from standard input

Description

1330 **xargs** is as specified in the Single UNIX Specification ISO POSIX (2003), but with differences as listed below.

Differences

1331 `-E`

1332 has unspecified behavior.

1333 `-I`

1334 has unspecified behavior.

1335 `-L`

1336 has unspecified behavior.

Notes

1337
1338 1. ~~Thus, applications should place options before operands, or use `—`, as needed. This text is needed because GNU~~
1339 ~~option parsing differs from POSIX. For example, `ls -a` in GNU `ls` means to list the current directory, showing all~~
1340 ~~files (that is, `.` is an operand and `-a` is an option). In POSIX, `.` and `-a` are both operands, and the command~~
1341 ~~means to list the current directory, and also the file named `-a`. Suggesting that applications rely on the setting of~~
1342 ~~the `POSIXLY_CORRECT` environment variable, or try to set it, seems worse than just asking the applications to~~
1343 ~~invoke commands in ways which work with either the POSIX or GNU behaviors.~~

1344 2. ~~Linux Standard Base~~

1345 3. ~~ISO/IEC 9945:2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3~~

1346 1. ~~The LSB generally does not include software development utilities nor does it specify `.o` and `.a` file formats.~~

1347 1. ~~The following two options are expected to be added in a future version of the LSB:~~

1348 ~~`-o office`~~

1349 ~~`—` sets the user's office room number.~~

1350 ~~`-p office phone`~~

1351 ~~`—` sets the user's office phone number.~~

1352 Note that some implementations contain a `"o other"` option which specifies an additional field called "other". Traditionally,
1353 this field is not subject to the constraints about legitimate characters in fields. Also, one traditionally shall have appropriate
1354 privileges to change the other field. At this point there is no consensus about whether it is desirable to specify the other field;
1355 applications may wish to avoid using it.

The "w work_phone" field found in some implementations should be replaced by the "p office_phone" field. The "r room_number" field found in some implementations is the equivalent of the "o office" option mentioned above; which one of these two options to specify will depend on implementation experience and the decision regarding the other field.

The intention is for chfn to match the behavior of finger; some historical implementations have been broken in the sense that finger and chfn do not agree on what the fields are.

1. The behavior specified here is similar to that specified by the Single UNIX Specification version 3 without the XSI option. However, the LSB forbids all options and the latter forbids only -n.

1. Need further investigation on the behavior of various implementations concerning whether program is a full pathname, the basename only, the program as named by argv[0], or what.

IV. Execution Environment

Chapter 4. File System Hierarchy

An LSB conforming implementation shall adhere to provide the mandatory portions of the filesystem hierarchy specified in the Filesystem Hierarchy Standard (FHS) 2.3.

~~An LSB conforming application shall follow the FHS.~~

(FHS), together with any additional requirements made in this specification.

An LSB conforming application shall conform to the Filesystem Hierarchy Standard.

The FHS allows many components or subsystems to be optional. An application shall check for the existence of an optional component before using it, and should behave in a reasonable manner if the optional component is not present.

The FHS requirement to locate the operating system kernel in either `/` or `/boot` does not apply if the operating system kernel does not exist as a file in the filesystem.

The FHS specifies certain behaviors for a variety of commands if they are present (for example, `ping` or `python`).

However, LSB applications shall not rely on any commands beyond those specified by the LSB. The mere existence of a command may not be used as an indication that the command behaves in any particular way.

The following directories or links need not be present: `/etc/X11` `/usr/bin/X11` `/usr/lib/X11` `/proc`

4.1. `/dev`

The following shall exist under `/dev`. Other devices may also exist in `/dev`. Device names may exist as symbolic links to other device nodes located in `/dev` or subdirectories of `/dev`. There is no requirement concerning major/minor number values.

`/dev/null`

An infinite data source and data sink. Data written to this device shall be discarded. Reads from this device shall always return end-of-file (EOF).

`/dev/zero`

This device is a source of zeroed out data. All data written to this device shall be discarded. A read from this device shall always return the requested number of bytes, each initialized to the value `'\0'`.

`/dev/tty`

In each process, a synonym for the controlling terminal associated with the process group of that process, if any. All reads and writes to this device shall behave as if the actual controlling terminal device had been opened.

Chapter 5. Additional Recommendations

5.1. Minimal granted Directory and File permissions

In this Chapter "System" means an "LSB conforming implementation" and "application" means an "LSB conforming (third party vendor) application".

The system shall grant to the application read and execute permissions on files needed to use all system interfaces (ABIs) mentioned in required by the LSB document and included standards specification.

5.2. Recommendations for applications on ownership and permissions

5.2.1. Directory Write Permissions

The application should not depend on having directory write permission outside `/tmp`, `/var/tmp`, invoking user's home directory and `/var/opt/package`, (where *package* is the name of the application package).

The application should not depend on owning these directories.

For these directories the application should be able to work with directory write permissions restricted by the `S_ISVTX` bit (otherwise known as the "sticky bit". (Which prevents the application from removing files owned by another user. This is classically done with `/tmp`, to prevent accidental deletion of "foreign" files.)).

5.2.2. File Write Permissions

The application should not depend on file write permission on files not owned by the user it runs under with the exception of its personal inbox `/var/mail/username`.

5.2.3. File Read and execute Permissions

The application should not depend on having read permission to every file and directory.

5.2.4. Suid and Sgid Permissions

The application should not depend on the `suid/sgid` set user ID or set group ID (the `S_ISUID` or `S_ISGID` permissions of a file not packaged with the application. Instead, the distribution is responsible for assuming that all system commands have the required permissions and work correctly.

Rationale: Let us make

In order to implement common security officers happy. Let's give them the freedom to take `sgid/suid` perms away, as long as they do not break policies it is strongly advisable for applications to use the system's functionality minimum set of security attributes necessary for correct operation. Applications that require substantial appropriate privilege are likely to cause problems with such security policies.

5.2.5. Privileged users

~~"Normal"~~In general, applications should not depend on running as a privileged user.

~~Special applications that have a reason to run under a privileged user, should outline these reasons clearly in their documentation, if they are not obvious as in~~ This specification uses the easeterm "appropriate privilege" throughout to identify operations that cannot be achieved without some special granting of a backup/restore programadditional privilege.

Applications that have a reason to run with appropriate privilege should outline this reason clearly in their documentation. Users of the application should be informed, that "this application demands security privileges, which could interfere with system security".

The application should not contain binary-only software that requires being run as rootwith appropriate privilege, as this makes security auditing harder or even impossible.

5.2.6. Changing permissions

~~The application should~~shall not change permissions of files and directories that do not belong to its own package. ~~To do so without~~Should an application require that certain files and directories not directly belonging to the package have a warning notice in the documentationparticular ownership, the application shall document this requirement, and may fail during installation if the permissions on these files is regarded as unfriendly actinappropriate.

5.2.7. Removable Media (Cdrom, Floppy, etc.)

Applications that expect to be runnable from removable media should not depend on logging in as a privileged user, and should be prepared to deal with a restrictive environment. Examples of such restrictions could be default mount options that disable set-user/group-ID attributes, disabling block or character-special files on the medium, or remapping the user/ and group IDs of files away from 0-⁺any privileged value.

Rationale

System vendors and local system administrators want to run applications from removable media, but want the possibility to control what the application can do.

5.2.8. Installable applications

~~If the installation of an application requires the execution of programs with superuser privileges, such programs should also be supplied in a human-readable form.~~

Where the installation of an application needs additional privileges, it must clearly document all files and system databases that are modified outside of those in `/opt/pkg-name` and `/var/opt/pkg-name`, other than those that may be updated by system logging or auditing activities.

Without this, the local system administrator would have to blindly trust a piece of software, particularly with respect to its security.

Notes

- ~~1. Rationale: System vendors and local system administrators want to run applications from removable media, but want the possibility to control what the application can do.~~

Chapter 6. Additional Behaviors

6.1. Mandatory Optional Behaviors

This section specifies behaviors in which there is optional behavior in one of the standards on which the LSB relies, and where the LSB requires a specific behavior. ⁺

The LSB does not require the kernel to be Linux; the set of mandated options reflects current existing practice, but may be modified in future releases.

LSB conforming implementations shall support the following options defined within the *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3*:

_POSIX_FSYNC
_POSIX_MAPPED_FILES
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MEMORY_PROTECTION
_POSIX_PRIORITY_SCHEDULING
_POSIX_REALTIME_SIGNALS
_POSIX_THREAD_ATTR_STACKADDR
_POSIX_THREAD_ATTR_STACKSIZE
_POSIX_THREAD_PROCESS_SHARED
_POSIX_THREAD_SAFE_FUNCTIONS
_POSIX_THREADS
_XOPEN_UNIX

The `opendir()` function shall consume a file descriptor in the same fashion as `open`, and therefore may fail with `EMFILE` or `ENFILE`.

The `START` and `STOP` `termios` characters shall be changeable, as described as optional behavior in the "General Terminal Interface" section of the *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3*.

The `access()` function shall fail with `errno` set to `EINVAL` if the `amode` argument contains bits other than those set by the bitwise inclusive OR of `R_OK`, `W_OK`, `X_OK` and `F_OK`.

The `link()` function shall require access to the existing file in order to succeed, as described as optional behavior in the *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3*.

Calling `unlink()` on a directory shall fail. Calling `link()` specifying a directory as the first argument shall fail. See also `unlink`. ²

Linux allows `rename()` on a directory without having write access, but the LSB does not require this.

6.1.1. Special Requirements

LSB conforming systems shall enforce certain special additional restrictions above and beyond those required by *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3*. ³

These additional restrictions are required in order to support the testing and certification programs associated with the LSB. In each case, these are values that defined macros must not have; conforming applications that use these values shall trigger a failure in the interface that is otherwise described as a "may fail".

The `fcntl()` function shall treat the "cmd" value -1 as invalid.

The "whence" value -1 shall be an invalid value for the `lseek()`, `fseek()` and `fcntl()` functions.

The value "-5" shall be an invalid signal number.

If the `sigaddset()` or `sigdelset()` functions are passed an invalid signal number, they shall return with `EINVAL`. Implementations are only required to enforce this requirement for signal numbers which are specified to be invalid by this specification (such as the -5 mentioned above).

The mode value "-1" to the `access()` function shall be treated as invalid.

A value of -1 shall be an invalid "_PC..." value for `pathconf()`.

A value of -1 shall be an invalid "_SC..." value for `sysconf()`.

The `nl_item` value "-1" shall be invalid for `nl_langinfo`.

The value -1 shall be an invalid "_CS..." value for `confstr()`.

The value "z" shall be an invalid *mode* argument to `popen()`.

Notes

~~1. The LSB does not require the kernel to be Linux; the set of mandated options reflects current existing practice, but may be modified in future releases.~~

~~2. Linux allows `rename()` on a directory without having write access, but the LSB does not require this.~~

~~3. These additional restrictions are required in order to support the testing and certification programs associated with the LSB. In each case, these are values that defined macros must not have; conforming applications that use these values shall trigger a failure in the interface that is otherwise described as a "may fail".~~

Chapter 7. Localization

Applications may either install a message catalog in the MO installation procedure shall supply the message catalog in a format as specified readable by the info page in version 0.10.40 of the gettext source package, or the application may execute the **msgfmt** command during its installation utility, which shall be invoked to compile the message catalog. In either case into an appropriate binary format on the target system.

Rationale

The original intent was to allow an application to contain the binary GNU MO format files. However, the format of these files is not officially stable, hence it is necessary to compile these catalogs on the target system. These binary catalogs may differ from architecture to architecture as well.

The resulting output binary message catalog shall be located in the package's private area under /opt, and the application may use `bindtextdomain()` to specify this location.

Implementations shall support the POSIX and C locales as specified in the ~~Single UNIX Specification~~ISO POSIX (2003).

7.1. Regular Expressions

Utilities that process regular expressions shall support Basic Regular Expressions and Extended Regular Expressions as specified in the ~~Single UNIX Specification~~ISO POSIX (2003), with the following exceptions:

Range expression (such as `[a-z]`) can be based on code point order instead of collating element order.

Equivalence class expression (such as `[=a=]`) and multi-character collating element expression (such as `[.ch.]`) are optional.

Handling of a multi-character collating element is optional.

This affects at least the following utilities: **grep** (`grep>`) (including **egrep**), **sed** (`sed>`), and **awk** (`awk>`).

7.2. ~~Filename Globbing~~Pattern Matching Notation

Utilities that perform filename ~~globbing~~pattern matching (also known as ~~Pattern Matching Notation~~Filename Globbing) shall do it as specified in the ~~Single UNIX Specification~~ISO POSIX (2003), Pattern Matching Notation, with the following exceptions:

~~Range expression~~Pattern bracket expressions (such as `[a-z]`) can be based on code point order instead of collating element order.

Equivalence class expression (such as `[=a=]`) and multi-character collating element expression (such as `[.ch.]`) are optional.

Handling of a multi-character collating element is optional.

This affects at least the following utilities: **cpio** (`cpio>`), **find** (`find>`), **ls** (`ls>`) and **tar** (`tar>`).

V. System Initialization

Chapter 8. System Initialization

8.1. Cron Jobs

~~Packages may not touch~~ In addition to the configuration file `/etc/crontab`, nor may they modify the individual user crontab files inspecified by ISO POSIX (2003) stored under `/var/spool/cron/crontabs`, the process that executes scheduled commands shall also process the following additional crontab files: `/etc/crontab`, `/etc/cron.d/*` The installation of a package shall not modify the configuration file `/etc/crontab`.

If a package ~~wants~~ wishes to install a job that has to be executed ~~via cron~~ periodically, it shall place a file in one of the following directories:

```
/etc/cron.daily  
/etc/cron.weekly  
/etc/cron.monthly  
/etc/cron.daily  
/etc/cron.weekly  
/etc/cron.monthly
```

As these directory names ~~says~~ suggest, the files within them are executed on a daily, weekly, or monthly basis, respectively, under the control of an entry in one of the system crontab files. See below for the rules concerning the names of ~~these files~~ in these directories.

~~If a certain job has to be executed more frequently than daily, the package shall install a file~~
~~/etc/cron.d/cron-name~~ tagged as configuration file. This file uses the same syntax as `/etc/crontab` and is processed by cron automatically.

It is recommended that files installed in any of these directories be scripts (e.g. shell scripts, Perl scripts, etc.) so that they may be modified by the local system administrator. ~~In addition, they shall be registered as configuration file.~~

The scripts in these directories ~~have to~~ should check, if all necessary programs are installed before they try to execute them. Otherwise, problems will arise ~~when~~ if a package ~~was~~ is removed (but not purged), since the configuration files are kept on the system in this situation.

If a certain job has to be executed at a different frequency (e.g. more frequently than daily), the package shall install a file `/etc/cron.d/cron-name` tagged as a configuration file. This file uses the same syntax as `/etc/crontab` and is processed by the system automatically.

To avoid namespace conflicts in the `/etc/cron.*` directories, the filenames used by LSB-compliant packages in `/etc/cron.daily`, `/etc/cron.weekly`, `/etc/cron.monthly`, or `/etc/cron.d` shall come from a managed namespace. These filenames may be assigned using one of the following methods:

- Assigned namespace. This namespace consists of names which only use the character set `[a-z0-9]`. In order to avoid conflicts these cron script names shall be reserved through the Linux Assigned Names and Numbers Authority (LANANA). Information about the LANANA may be found at www.lanana.org (<http://www.lanana.org>).

Commonly used names shall be reserved in advance; developers for projects should be encouraged reserve names from LANANA, so that each distribution can use the same name, and to avoid conflicts with other projects.

- Hierarchical namespace. This namespace consists of ~~scripts names which look like this~~ script names of the form: [hier1]-[hier2]-...-[name], where name is again taken from the character set [a-z0-9], and where there may be one or more [hier-n] components. [hier1] may either be an LSB provider name assigned by the LANANA, or it may be owners' DNS name in lower case, with at least one '.'. I.e.:g. "debian.org", "staroffice.sun.com", etc. The LSB provider name assigned by LANANA shall only consist of the ASCII characters [a-z0-9].
- Reserved namespace. This namespace consists of script names which begin with the character '_', and is reserved for distribution use only. This namespace should be used for core packages only, ~~and in general use of this namespace is highly discouraged.~~

8.2. Init Script Actions

Init files provided by LSB applications shall accept one argument, saying what to do:

start	start the service
stop	stop the service
restart	stop and restart the service if the service is already running, otherwise start the service
try-restart	restart the service if the service is already running
reload	cause the configuration of the service to be reloaded without actually stopping and restarting the service
force-reload	cause the configuration to be reloaded if the service supports this, otherwise restart the service if it is running
status	print the current status of the service

The start, stop, restart, force-reload, and status commands shall be supported by all init files; the reload and the try-restart options are optional. Other init script actions may be defined by the init script.

Init files shall ensure that they will behave sensibly if invoked with start when the service is already running, or with stop when it isn't, and that they don't kill unfortunately-named user processes. The best way to achieve this is to use the init-script functions provided by /lib/lsb/init-functions.

If a service reloads its configuration automatically (as in the case of cron, for example), the reload option of the init file shall behave as if the configuration has been reloaded successfully. The restart, try-restart, reload and force-reload action may be atomic; i.e. if a service is known not be operational after a restart or reload, the script may return an error without any further action.

These executable files shall not fail obscurely when the configuration files remain but the package has been removed, as the default in [the packaging system] is to leave configuration files on the system after the package has been removed. Only when it is executed with the [purge] option will [the packaging system] remove configuration files. Therefore, you should include a test statement at the top of the file, like this:

```
test -f program-executed-later-in-file || exit 5
```

or take the equivalent action if the init file is not a shell script.

If the status command is given, the init script will return the following exit status codes.

0	program is running or service is OK
1	program is dead and /var/run pid file exists
2	program is dead and /var/lock lock file exists
3	program is not running

4 program or service status is unknown
 5-99 reserved for future LSB use
 100-149 reserved for distribution use
 150-199 reserved for application use
 61 200-254 reserved

62 In the case of init script commands other than "status" (i.e., "start", "stop", "restart", "try-restart", "reload", and
 63 "force-reload"), the init script shall return an exit status of zero if the action described by the argument has been
 64 successful. Otherwise, the exit status shall be non-zero, as defined below. In addition to straightforward success, the
 65 following situations are also to be considered successful:

- 66 • restarting a service (instead of reloading it) with the "force-reload" argument
- 67 • running "start" on a service already running
- 68 • running "stop" on a service already stopped or not running
- 69 • running "restart" on a service already stopped or not running
- 70 • running "try-restart" on a service already stopped or not running

71 In case of an error, while processing any init script action except for "status", the init script shall print an error message
 72 and return one of the following non-zero exit status codes.

1 generic or unspecified error (current practice)
 2 invalid or excess argument(s)
 3 unimplemented feature (for example, "reload")
 4 user had insufficient privilege
 5 program is not installed
 6 program is not configured
 7 program is not running
 8-99 reserved for future LSB use
 100-149 reserved for distribution use
 150-199 reserved for application use
 73 200-254 reserved

74 Error and status messages should be printed with the logging functions such as `log_failure_msg` and so on. Scripts may
 75 write to standard error or standard output, but implementations need not present text written to standard error/output to
 76 the user or do anything else with it.

77 Since init files may be run manually by a system administrator with non-standard environment variable values for
 78 `PATH`, `USER`, `LOGNAME`, etc. init files shall not depend on the values of these environment variables. They should
 79 set them to some known/default values if they are needed.

8.3. Comment Conventions for Init Scripts

80 LSB applications which need to execute script(s) at bootup and/or shutdown may provide one or more `init.d` files.
 81 These files are installed by the `install_initd` program described below, which copies it into a standard directory and
 82 makes whatever other adjustments (creation of symlinks, creation of entries in a database, etc.) are necessary so that
 83 the script can be run at boot-time. ¹

84 In the `init.d` file, information about the shell script shall be delimited by the lines "### BEGIN INIT INFO" and "###
 85 END INIT INFO". These delimiter lines may contain trailing whitespace, which shall be ignored. Inside this block
 86 there shall be lines of the form "# {keyword}: [arg1] [arg2] ...". (All lines inside this block start with a hash (#))

character in the first column, so that shell treats them as comments.) There shall be exactly one space character between "#" and the keyword.² The following keywords, with their arguments are defined in this specification:

```

—— # Provides: boot_facility_1 [ boot_facility_2 ...]
      # Required-Start: boot_facility_1 [ boot_facility_2 ...]
      # Required-Stop: boot_facility_1 [ boot_facility_2 ...]
      # Should-Start: boot_facility_1 [ boot_facility_2 ...]
      # Should-Stop: boot_facility_1 [ boot_facility_2 ...]
      # Default-Start: run_level_1 [ run_level_2 ...]
      # Default-Stop: run_level_1 [ run_level_2 ...]
      # Short-Description: short_description
      # Description: multiline_description

```

Additional keywords may be defined in future LSB specifications. Distributions may define local extensions by using the prefix "X-[distribution name]" --- for example, "X-RedHat-foobardecl", or "X-Debian-xyzzycl".

An init.d shell script may declare using the "Required-Start:" header that it shall not be run until certain boot facilities are provided. This information is used by the installation tool or the boot-time boot-script execution facility to assure that init scripts are run in the correct order. When an init script is run with a "start" argument, the boot facility or facilities specified in the "Provides" header shall be considered present, and hence init scripts which require those boot facilities would then be eligible to be run. When an init script is run with a "stop" argument, the boot facilities specified in the "Provides" header are considered no longer present. There are naming conventions for boot facilities and system facilities, as described in a following section.

Similarly, the "Required-Stop:" header defines which facilities shall still be available during the shutdown of that service. Hence, the init script system should avoid stopping shell scripts which provide those facilities until this shell script is stopped.

The "Should-Start:" header defines which facilities if present should be started before this service. This allows for weak dependencies which do not cause the service to fail if a facility is not available. But may cause reduced functionality of the service. Compliant applications should not rely on the existence of this feature.

The "Should-Stop:" header defines which facilities should be still available during the shutdown of that service.

The "Default-Start" and "Default-Stop" headers define which run levels should by default run the script with a start or stop argument, respectively, to start or stop the services controlled by the init script.³

The "Short-Description" and "Description" header fields are used to provide text which describes the actions of the init script. The "short_description" shall be a relatively short, pithy description of the init script, where as the "multiline_description" can be a much longer piece of text that may span multiple lines. In a multiline description, each continuation line shall begin with a '#' followed by tab character or a '#' followed by at least two space characters. The multiline description is terminated by the first line that does not match this criteria.

The comment conventions described in this session are only required for use by LSB-compliant applications; system init scripts as provided by LSB-compliant run-time environments are *not* required to use the scheme outlined here.

8.4. Installation and Removal of init.d Files

An init.d file is installed in /etc/init.d (which may be a symlink to another location). This can be done by the package installer. See Script Names>. During the package's postinstall script, the program "/usr/lib/lsb/install_initd" configures the distribution's boot script system to call the package's init.d file at the appropriate time.⁴

The install_initd program takes a single argument, the pathname to the /etc/init.d file. For example:

```
127 /usr/lib/lsb/install_initd /etc/init.d/example.com-coffee
```

128 The install_initd program shall return an exit status of zero if the init.d file has been successfully installed or if the the
129 init.d file was already installed. If the required boot facilities cannot be fulfilled an exit status of one shall be returned
130 and the init.d file shall not be installed.

131 When a software package is removed, the package's preuninstall script shall call /usr/lib/lsb/remove_initd and pass the
132 pathname to the /etc/init.d file. The package manager is still responsible for removing the /etc/init.d file; the
133 remove_initd program is provided in case the distribution needs to clean up any other modifications in the
134 distribution's boot script system that might have been made by the install_initd program. For example:

```
135 /usr/lib/lsb/remove_initd /etc/init.d/example.com-coffee
```

136 The remove_initd program shall return an exit status of zero if the init.d file has been successfully removed or if the the
137 init.d file is not installed. If another init.d file which depends on a boot facility provided by this init.d file is installed,
138 an exit status of one shall be returned and the init.d file shall remain installed.

139 There should be a tool available to the user (e.g., RedHat's chkconfig) which can be used by the system administrator
140 to easily manipulate at which init levels a particular init.d script is started or stopped. This specification currently does
141 not specify such an interface, however.

8.5. Run Levels

142 The following run levels are specified for use by the "Default-Start:" and "Default-Stop:" specifiers as defined by the
143 section *Comment Conventions for Init Scripts*>. Many LSB run-time environments commonly use these run level
144 definitions, and in the absence of other considerations, providers of run-time environments are strongly encouraged to
145 follow this convention to provide consistency for system administrators who need to work with multiple distributions.
146 However, it is not required that LSB-compliant run-time environments use these run levels; the distribution-provided
147 install_initd script may map the run levels specified below to whatever distribution-specified run levels are most
148 appropriate.

0	halt
1	single user mode
2	multiuser with no network services exported
3	normal/full multiuser
4	reserved for local use, default is normal/full multiuser
5	multiuser with xdm or equivalent
6	reboot

8.6. Facility Names

150 Boot facilities are used to indicate dependencies in init scripts, as defined in a previous section. Facility names that
151 begin with a dollar sign ('\$') are system facility names, defined by the LSB, and SHALL be provided by distributions.
152 ⁵ LSB applications shall not provide facilities that begin with a dollar sign. This document defines the following
153 facility names:

\$local_fs	all local filesystems are mounted
\$network	low level networking (ethernet card; may imply PCMCIA running)
\$named	daemons which may provide hostname resolution (if

present) are running⁶
 \$portmap daemons providing SunRPC/ONCRPC portmapping
 service⁷ (if present) are running
 \$remote_fs all remote filesystems are mounted⁸.
 \$syslog system logger is operational
 154 \$time the system time has been set⁹
 155 Other (non-system) facilities may be defined by other LSB applications. These facilities shall be named using the
 156 same conventions defined for naming init.d script names. Commonly, the facility provided by an LSB application
 157 init.d script will have the same name as the name assigned to the init.d script.

8.7. Script Names

158 Since the init.d scripts shall live in a single directory, they shall come from a single namespace. Three means of
 159 assigning names from this namespace are available:

- 160 • Assigned namespace. This namespace consists of names which only use the character set [a-z0-9]. This space is
 161 desirable for scripts which system administrators may often wish to run manually: e.g., "/etc/init.d/named restart" In
 162 order to avoid conflicts these init.d names shall be reserved through the Linux Assigned Names and Numbers
 163 Authority (LANANA). Information about the LANANA may be found at www.lanana.org
 164 (<http://www.lanana.org>).
- 165 Commonly used names shall be reserved in advance; developers for projects should be encouraged to reserve names
 166 from LANANA, so that each distribution can use the same name, and to avoid conflicts with other projects.
- 167 • Hierarchical namespace. This namespace consists of scripts names which look like this: [hier1]-[hier2]-...-[name],
 168 where name is again taken the character set [a-z0-9], and where there may be one or more [hier-n] components.
 169 [hier1] may either be an LSB provider name assigned by the LANANA, or it may be owners' DNS name in lower
 170 case, with at least one '.' (e.g., "debian.org", "staroffice.sun.com"). The LSB provider name assigned by LANANA
 171 shall only consist of the ASCII characters [a-z0-9].
- 172 • Reserved namespace. This namespace consists of script names which begin with the character '_', and is reserved
 173 for distribution use only. This namespace should be used for core packages only, and in general use of this
 174 namespace is highly discouraged.

175 In general, if a package or some system function is likely to be used on multiple systems, the package developers or the
 176 distribution SHOULD get a registered name through LANANA, and distributions should strive to use the same name
 177 whenever possible. For applications which may not be "core" or may not be commonly installed, the hierarchical
 178 namespace may be more appropriate. An advantage to the hierarchical namespace is that there is no need to consult
 179 with the LANANA before obtaining an assigned name.

180 Short names are highly desirable, since many system administrators like to use them to manually start and stop
 181 services. Given this, they should be standardized on a per-package basis. This is the rationale behind having a
 182 LANANA organization to assign these names. The LANANA may be called upon to handle other namespace issues,
 183 such as package/prerequisites naming (which is essential to making prerequisites to work correctly).

8.8. Init Script Functions

184 Each LSB-compliant init.d script shall source the file `/lib/lsb/init-functions`. This file shall cause the
 185 following shell script commands to be defined. This can be done either by adding a directory to the PATH variable
 186 which defines these commands, or by defining sh aliases. While the distribution-provided aliases may choose to use

187	shell extensions (at the distribution's option), the LSB init.d files themselves should only depend in shell features as	
188	defined by the LSB.	
189	The start_daemon , killproc and pidofproc functions shall use this algorithm for determining the status and the pid(s)	
190	of the specified program. They shall read the pidfile specified or otherwise <code>/var/run/basename.pid</code> and use the	
191	pid(s) herein when determining whether a program is running. The method used to determine the status is imple-	
192	mentation defined, but should allow for non-binary programs. ¹⁰ Compliant implementations may use other mecha-	
193	nisms besides those based on pidfiles, unless the <code>-p pidfile</code> option has been used. Compliant applications should not	
194	rely on such mechanisms and should always use a pidfile. When a program is stopped, it should delete its pidfile.	
195	Multiple pid(s) shall be separated by a single space in the pidfile and in the output of pidofproc .	
	start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args]	<p>This runs the specified program as a daemon.</p> <p>start_daemon shall check if the program is already running using the algorithm given above. If so, it shall not start another copy of the daemon unless the <code>-f</code> option is given. The <code>-n</code> option specifies a nice level. See <code>nice(1)</code>. start_daemon should return the LSB defined exit status codes. It shall return 0 if the program has been successfully started or is running and not 0 otherwise.</p>
	killproc [-p pidfile] pathname [signal]	<p>This stops the specified program. The program is found using the algorithm given above. If a signal is specified, using the <code>-signal_name</code> or <code>-signal_number</code> syntaxes as specified by the kill command, the program is sent that signal. Otherwise, a SIGTERM followed by a SIGKILL after some number of seconds shall be sent. If a program has been terminated, the pidfile should be removed if the terminated process has not already done so. Compliant applications may use the basename instead of the pathname. killproc should return the LSB defined exit status codes. If called without a signal, it shall return 0 if the program has been stopped or is not running and not 0 otherwise. If a signal is given, it shall return 0 only if the program is running.</p>
	pidofproc [-p pidfile] pathname	<p>This function returns one or more pid(s) for a particular daemon using the algorithm given above. Only pids of running processes should be returned. Compliant applications may use the basename instead of the pathname. pidofproc should return the LSB defined exit status codes for "status". It shall return 0 if the program is running and not 0 otherwise.</p>
	log_success_msg "message"	<p>This requests the distribution to print a success message. The message should be relatively short; no more than 60 characters is highly desirable.</p>
	log_failure_msg "message"	<p>This requests the distribution to print a failure message. The message should be relatively short; no more than 60 characters is highly desirable.</p>
	log_warning_msg "message"	<p>This requests the distribution to print a warning message. The message should be relatively short; no more than 60 characters is highly desirable.</p>

Notes

1. This specification does not require, but is designed to allow, the development of a system which runs boot scripts in parallel. Hence, enforced-serialization of scripts is avoided unless it is explicitly necessary.
2. More than one space, or a tab character, indicates the continuation line.
3. For example, if you want a service to run in runlevels 3, 4, and 5 (only), specify "Default-Start: 3 4 5" and "Default-Stop: 0 1 2 6".
4. For example, **install_initd** might create symbolic links in /etc/rc2.d and other such directories which point to the files in /etc/init.d (or it might update a database, or some other mechanism). The init.d files themselves should already be in /etc/init.d before running **install_initd**.
5. The dollar sign does not indicate variable expansion as in many Linux utilities. Starting a facility name with a dollar sign is merely a way of dividing the namespace between the system and applications.
6. For example, daemons to query DNS, NIS+, or LDAP
7. as defined in RFC 1833
8. In some LSB run-time environments, filesystems such as /usr may be remote. Many applications that require \$local_fs will probably require also require \$remote_fs
9. i.e., using a network-based time program such as ntp or rdate, or via the hardware Real Time Clock
10. This note is only informative. Commonly used methods check either for the existence of the /proc/pid directory or use /proc/pid/exe and /proc/pid/cmdline. Relying only on /proc/pid/exe is discouraged since this results in a not-running status for daemons that are written in a script language.

VI. Users & Groups

Chapter 9. Users & Groups

A "user name" is a string that is used to identify a user. A "login name" is a user name that is associated with a system login. A "user id" is a non-negative integer, which can be contained in an object of type `uid_t`, that is used to identify a system user.

When the identity of a user is associated with a process, a user ID value is referred to as a real user ID, or an effective user ID. [POSIX 1003.1 1996]

A "group name" is a string that is used to identify a set of users. A "group id" is a non-negative integer, which can be contained in a object of type `gid_t`, that is used to identify a group of system users. Each system user is a member of at least one group. When the identity of a group is associated with a process, a group ID value is referred to as a real group ID, or an effective group ID. [POSIX 1003.1 1996]

9.1. User and Group Database

The format of the User and Group databases is not specified. Programs may only read these databases using the provided API. Changes to these databases should be made using the provided commands.

9.2. User & Group Names

Below is a table of required mnemonic user and group names. This specification makes no attempt to numerically assign uid or gid numbers. The exception is the uid and gid for "root" which are equal to 0.

Table 9-1. Required User & Group Names

User	Group	Comments
root	root	Administrative user with no restrictions all appropriate privileges
bin	bin	Legacy UID/GID ^a
daemon	daemon	Legacy UID/GID ^b

Notes:

- The 'bin' UID/GID is included for compatibility with legacy applications. New applications should no longer use the 'bin' UID/GID.
- The 'daemon' UID/GID was used as an unprivileged UID/GID for daemons to execute under in order to limit their access to the system. Generally daemons should now run under individual UID/GIDs in order to further partition daemons from one another.

Below is a table of optional mnemonic user and group names. This specification makes no attempt to numerically assign uid or gid numbers. If the username exists on a system, then they should be in the suggested corresponding group. These user and group names are for use by distributions, not by applications.

Table 9-2. Optional User & Group Names

User	Group	Comments
adm	adm	Administrative special privileges
lp	lp	Printer special privileges
sync	sync	Login to sync the system
shutdown	shutdown	Login to shutdown the system
halt	halt	Login to halt the system
mail	mail	Mail special privileges
news	news	News special privileges
uucp	uucp	UUCP special privileges
operator	root	Operator special privileges
man	man	Man special privileges
nobody	nobody	Used by NFS

The differences in numeric values of the uids and gids between systems on a network can be reconciled via NIS, rdist(1), rsync(1), or ugid(8). Only a minimum working set of "user names" and their corresponding "user groups" are required. Applications cannot assume non system user or group names will be defined.

Applications cannot assume any policy for the default umask or the default directory permissions a user may have. Applications should enforce user only file permissions on private files such as mailboxes. The location of the users home directory is also not defined by policy other than the recommendations of the FHS and shall be obtained by the *pwnam(3) calls.

9.3. UID Ranges

The system UIDs from 0 to 99 should be statically allocated by the system, and shall not be created by applications.

The system UIDs from 100 to 499 should be reserved for dynamically allocation by system administrators and post install scripts using useradd(1).

9.4. Rationale

The purpose of specifying optional users and groups is to reduce the potential for name conflicts between applications and distributions.

Appendix A. Alphabetical Listing of Interfaces

A.1. libX11libc

The behaviour of the interfaces in this library is specified by the following Standards.

Linux Standard BaseLarge File Support
this specification

Table A-1. libX11 Function Interfaces

XActivateScreenSaver[1]	XIconifyWindow[1]	XcmsCIELabQueryMinL[1]
XAddConnectionWatch[1]	XIfEvent[1]	XcmsCIELabToCIEXYZ[1]
XAddExtension[1]	XImageByteOrder[1]	XcmsCIELabWhiteShiftColors[1]
XAddHost[1]	XInitExtension[1]	XcmsCIELuvClipL[1]
XAddHosts[1]	XInitImage[1]	XcmsCIELuvClipLuv[1]
XAddPixel[1]	XInitThreads[1]	XcmsCIELuvClipuv[1]
XAddToExtensionList[1]	XInsertModifiermapEntry[1]	XcmsCIELuvQueryMaxC[1]
XAddToSaveSet[1]	XInstallColormap[1]	XcmsCIELuvQueryMaxL[1]
XAllPlanes[1]	XInternAtom[1]	XcmsCIELuvQueryMaxLC[1]
XAllocClassHint[1]	XInternAtoms[1]	XcmsCIELuvQueryMinL[1]
XAllocColor[1]	XInternalConnectionNumbers[1]	XcmsCIELuvToCIEnvY[1]
XAllocColorCells[1]	XIntersectRegion[1]	XcmsCIELuvWhiteShiftColors[1]
XAllocColorPlanes[1]	XKeycodeToKeysym[1]	XcmsCIEXYZToCIELab[1]
XAllocIconSize[1]	XKeysymToKeycode[1]	XcmsCIEXYZToCIEnvY[1]
XAllocNamedColor[1]	XKeysymToString[1]	XcmsCIEXYZToCIExyY[1]
XAllocSizeHints[1]	XKillClient[1]	XcmsCIEXYZToRGBi[1]
XAllocStandardColormap[1]	XLastKnownRequestProcessed[1]	XcmsCIEnvYToCIELuv[1]
XAllocWMHints[1]	XListDepths[1]	XcmsCIEnvYToCIEXYZ[1]
XAllowEvents[1]	XListExtensions[1]	XcmsCIEnvYToTekHVC[1]
XAutoRepeatOff[1]	XListFonts[1]	XcmsCIExyYToCIEXYZ[1]
XAutoRepeatOn[1]	XListFontsWithInfo[1]	XcmsClientWhitePointOfCCC[1]
XBaseFontNameListOfFontSet[1]	XListHosts[1]	XcmsConvertColors[1]
XBell[1]	XListInstalledColormaps[1]	XcmsCreateCCC[1]

XBitmapBitOrder[1]	XListPixmapFormats[1]	XcmsDefaultCCC[1]
XBitmapPad[1]	XListProperties[1]	XcmsDisplayOfCCC[1]
XBitmapUnit[1]	XLoadFont[1]	XcmsFormatOfPrefix[1]
XBlackPixel[1]	XLoadQueryFont[1]	XcmsFreeCCC[1]
XBlackPixelOfScreen[1]	XLocaleOfFontSet[1]	XcmsLookupColor[1]
XCellsOfScreen[1]	XLocaleOfIM[1]	XcmsPrefixOfFormat[1]
XChangeActivePointerGrab[1]	XLocaleOfOM[1]	XcmsQueryBlack[1]
XChangeGC[1]	XLockDisplay[1]	XcmsQueryBlue[1]
XChangeKeyboardControl[1]	XLookupColor[1]	XcmsQueryColor[1]
XChangeKeyboardMapping[1]	XLookupKeysym[1]	XcmsQueryColors[1]
XChangePointerControl[1]	XLookupString[1]	XcmsQueryGreen[1]
XChangeProperty[1]	XLowerWindow[1]	XcmsQueryRed[1]
XChangeSaveSet[1]	XMapRaised[1]	XcmsQueryWhite[1]
XChangeWindowAttributes[1]	XMapSubwindows[1]	XcmsRGBToRGBi[1]
XCheckIfEvent[1]	XMapWindow[1]	XcmsRGBiToCIEXYZ[1]
XCheckMaskEvent[1]	XMaskEvent[1]	XcmsRGBiToRGB[1]
XCheckTypedEvent[1]	XMatchVisualInfo[1]	XcmsScreenNumberOfCCC[1]
XCheckTypedWindowEvent[1]	XMaxCmapsOfScreen[1]	XcmsScreenWhitePointOfCCC[1]
XCheckWindowEvent[1]	XMaxRequestSize[1]	XcmsSetCCCOFColormap[1]
XCirculateSubwindows[1]	XMinCmapsOfScreen[1]	XcmsSetCompressionProc[1]
XCirculateSubwindowsDown[1]	XMoveResizeWindow[1]	XcmsSetWhiteAdjustProc[1]
XCirculateSubwindowsUp[1]	XMoveWindow[1]	XcmsSetWhitePoint[1]
XClearArea[1]	XNewModifiermap[1]	XcmsStoreColor[1]
XClearWindow[1]	XNextEvent[1]	XcmsStoreColors[1]
XClipBox[1]	XNextRequest[1]	XcmsTekHVCClipC[1]
XCloseDisplay[1]	XNoOp[1]	XcmsTekHVCClipV[1]
XCloseIM[1]	XOMOFOC[1]	XcmsTekHVCClipVC[1]
XCloseOM[1]	XOffsetRegion[1]	XcmsTekHVCQueryMaxC[1]
XConfigureWindow[1]	XOpenDisplay[1]	XcmsTekHVCQueryMaxV[1]
XConnectionNumber[1]	XOpenIM[1]	XcmsTekHVCQueryMaxVC[1]
XContextDependentDrawing[1]	XOpenOM[1]	XcmsTekHVCQueryMaxVSamples

		{1}
XContextualDrawing[1]	XParseColor[1]	XemsTekHVCQueryMinV[1]
XConvertCase[1]	XParseGeometry[1]	XemsTekHVCToCIEuvY[1]
XConvertSelection[1]	XPeekEvent[1]	XemsTekHVCWhiteShiftColors[1]
XCopyArea[1]	XPeekIfEvent[1]	XemsVisualOfCCC[1]
XCopyColormapAndFree[1]	XPending[1]	XkbAllocClientMap[1]
XCopyGC[1]	XPlanesOfScreen[1]	XkbAllocCompatMap[1]
XCopyPlane[1]	XPointInRegion[1]	XkbAllocControls[1]
XCreateBitmapFromData[1]	XPolygonRegion[1]	XkbAllocGeomColors[1]
XCreateColormap[1]	XProcessInternalConnection[1]	XkbAllocGeomDoodads[1]
XCreateFontCursor[1]	XProtocolRevision[1]	XkbAllocGeomKeyAliases[1]
XCreateFontSet[1]	XProtocolVersion[1]	XkbAllocGeomKeys[1]
XCreateGC[1]	XPutBackEvent[1]	XkbAllocGeomOutlines[1]
XCreateGlyphCursor[1]	XPutImage[1]	XkbAllocGeomOverlayKeys[1]
XCreateIC[1]	XPutPixel[1]	XkbAllocGeomOverlayRows[1]
XCreateImage[1]	XQLength[1]	XkbAllocGeomOverlays[1]
XCreateOC[1]	XQueryBestCursor[1]	XkbAllocGeomPoints[1]
XCreatePixmap[1]	XQueryBestSize[1]	XkbAllocGeomProps[1]
XCreatePixmapCursor[1]	XQueryBestStipple[1]	XkbAllocGeomRows[1]
XCreatePixmapFromBitmapData[1 †	XQueryBestTile[1]	XkbAllocGeomSectionDoodads[1]
XCreateRegion[1]	XQueryColor[1]	XkbAllocGeomSections[1]
XCreateSimpleWindow[1]	XQueryColors[1]	XkbAllocGeomShapes[1]
XCreateWindow[1]	XQueryExtension[1]	XkbAllocGeometry[1]
XDefaultColormap[1]	XQueryFont[1]	XkbAllocIndicatorMaps[1]
XDefaultColormapOfScreen[1]	XQueryKeymap[1]	XkbAllocKeyboard[1]
XDefaultDepth[1]	XQueryPointer[1]	XkbAllocNames[1]
XDefaultDepthOfScreen[1]	XQueryTextExtents[1]	XkbAllocServerMap[1]
XDefaultGC[1]	XQueryTextExtents16[1]	XkbApplyCompatMapToKey[1]
XDefaultGCOfScreen[1]	XQueryTree[1]	XkbBell[1]
XDefaultRootWindow[1]	XRaiseWindow[1]	XkbBellEvent[1]

XDefaultScreen[1]	XReadBitmapFile[1]	XkbChangeEnabledControls[1]
XDefaultScreenOfDisplay[1]	XReadBitmapFileData[1]	XkbChangeMap[1]
XDefaultString[1]	XRebindKeysym[1]	XkbChangeNames[1]
XDefaultVisual[1]	XRecolorCursor[1]	XkbChangeTypesOfKey[1]
XDefaultVisualOfScreen[1]	XReconfigureWMWindow[1]	XkbComputeEffectiveMap[1]
XDefineCursor[1]	XRectInRegion[1]	XkbComputeRowBounds[1]
XDeleteContext[1]	XRefreshKeyboardMapping[1]	XkbComputeSectionBounds[1]
XDeleteModifiermapEntry[1]	XRegisterIMInstantiateCallback[1]	XkbComputeShapeBounds[1]
XDeleteProperty[1]	XRemoveConnectionWatch[1]	XkbComputeShapeTop[1]
XDestroyIC[1]	XRemoveFromSaveSet[1]	XkbCopyKeyType[1]
XDestroyImage[1]	XRemoveHost[1]	XkbCopyKeyTypes[1]
XDestroyOC[1]	XRemoveHosts[1]	XkbFindOverlayForKey[1]
XDestroyRegion[1]	XReparentWindow[1]	XkbForceBell[1]
XDestroySubwindows[1]	XResetScreenSaver[1]	XkbFreeClientMap[1]
XDestroyWindow[1]	XResizeWindow[1]	XkbFreeCompatMap[1]
XDirectionalDependentDrawing[1]	XResourceManagerString[1]	XkbFreeComponentList[1]
XDisableAccessControl[1]	XRestackWindows[1]	XkbFreeControls[1]
XDisplayCells[1]	XRootWindow[1]	XkbFreeGeomColors[1]
XDisplayHeight[1]	XRootWindowOfScreen[1]	XkbFreeGeomDoodads[1]
XDisplayHeightMM[1]	XRotateBuffers[1]	XkbFreeGeomKeyAliases[1]
XDisplayKeycodes[1]	XRotateWindowProperties[1]	XkbFreeGeomKeys[1]
XDisplayMotionBufferSize[1]	XSaveContext[1]	XkbFreeGeomOutlines[1]
XDisplayName[1]	XScreenCount[1]	XkbFreeGeomOverlayKeys[1]
XDisplayOfHM[1]	XScreenNumberOfScreen[1]	XkbFreeGeomOverlayRows[1]
XDisplayOfOM[1]	XScreenOfDisplay[1]	XkbFreeGeomOverlays[1]
XDisplayOfScreen[1]	XScreenResourceString[1]	XkbFreeGeomPoints[1]
XDisplayPlanes[1]	XSelectInput[1]	XkbFreeGeomProperties[1]
XDisplayString[1]	XSendEvent[1]	XkbFreeGeomRows[1]
XDisplayWidth[1]	XServerVendor[1]	XkbFreeGeomSections[1]
XDisplayWidthMM[1]	XSetAccessControl[1]	XkbFreeGeomShapes[1]
XDoesBackingStore[1]	XSetArcMode[1]	XkbFreeGeometry[1]

XDoesSaveUnders[1]	XSetAuthorization[1]	XkbFreeIndicatorMaps[1]
XDrawArc[1]	XSetBackground[1]	XkbFreeKeyboard[1]
XDrawArcs[1]	XSetClassHint[1]	XkbFreeNames[1]
XDrawImageString[1]	XSetClipMask[1]	XkbFreeServerMap[1]
XDrawImageString16[1]	XSetClipOrigin[1]	XkbGetAutoRepeatRate[1]
XDrawLine[1]	XSetClipRectangles[1]	XkbGetCompatMap[1]
XDrawLines[1]	XSetCloseDownMode[1]	XkbGetControls[1]
XDrawPoint[1]	XSetCommand[1]	XkbGetGeometry[1]
XDrawPoints[1]	XSetDashes[1]	XkbGetIndicatorMap[1]
XDrawRectangle[1]	XSetErrorHandler[1]	XkbGetIndicatorState[1]
XDrawRectangles[1]	XSetFillRule[1]	XkbGetKeyActions[1]
XDrawSegments[1]	XSetFillStyle[1]	XkbGetKeyBehaviors[1]
XDrawString[1]	XSetFont[1]	XkbGetKeyExplicitComponents[1]
XDrawString16[1]	XSetFontPath[1]	XkbGetKeyModifierMap[1]
XDrawText[1]	XSetForeground[1]	XkbGetKeySyms[1]
XDrawText16[1]	XSetFunction[1]	XkbGetKeyTypes[1]
XEHeadOfExtensionList[1]	XSetGraphicsExposures[1]	XkbGetKeyboard[1]
XESetBeforeFlush[1]	XSetICFocus[1]	XkbGetKeyboardByName[1]
XESetCloseDisplay[1]	XSetICValues[1]	XkbGetMap[1]
XESetCopyGC[1]	XSetIMValues[1]	XkbGetMapChanges[1]
XESetCreateFont[1]	XSetIOErrorHandler[1]	XkbGetNamedGeometry[1]
XESetCreateGC[1]	XSetIconName[1]	XkbGetNamedIndicator[1]
XESetError[1]	XSetIconSizes[1]	XkbGetNames[1]
XESetErrorString[1]	XSetInputFocus[1]	XkbGetState[1]
XESetEventToWire[1]	XSetLineAttributes[1]	XkbGetUpdatedMap[1]
XESetFlushGC[1]	XSetLocaleModifiers[1]	XkbGetVirtualMods[1]
XESetFreeFont[1]	XSetModifierMapping[1]	XkbGetXlibControls[1]
XESetFreeGC[1]	XSetNormalHints[1]	XkbIgnoreExtension[1]
XESetPrintErrorValues[1]	XSetOCValues[1]	XkbInitCanonicalKeyTypes[1]
XESetWireToError[1]	XSetOMValues[1]	XkbKeyTypesForCoreSymbols[1]
XESetWireToEvent[1]	XSetPlaneMask[1]	XkbKeycodeToKeysym[1]

XEmptyRegion[1]	XSetPointerMapping[1]	XkbKeysymToModifiers[1]
XEnableAccessControl[1]	XSetRGBColormaps[1]	XkbLatchGroup[1]
XEqualRegion[1]	XSetRegion[1]	XkbLatchModifiers[1]
XEventMaskOfScreen[1]	XSetScreenSaver[1]	XkbLibraryVersion[1]
XEventsQueued[1]	XSetSelectionOwner[1]	XkbListComponents[1]
XExtendedMaxRequestSize[1]	XSetSizeHints[1]	XkbLockGroup[1]
XExtentsOfFontSet[1]	XSetStandardColormap[1]	XkbLockModifiers[1]
XFetchBuffer[1]	XSetStandardProperties[1]	XkbLookupKeyBinding[1]
XFetchBytes[1]	XSetState[1]	XkbLookupKeySym[1]
XFetchName[1]	XSetStipple[1]	XkbNoteControlsChanges[1]
XFillArc[1]	XSetSubwindowMode[1]	XkbNoteMapChanges[1]
XFillAres[1]	XSetTSOrigin[1]	XkbNoteNameChanges[1]
XFillPolygon[1]	XSetTextProperty[1]	XkbOpenDisplay[1]
XFillRectangle[1]	XSetTile[1]	XkbQueryExtension[1]
XFillRectangles[1]	XSetTransientForHint[1]	XkbRefreshKeyboardMapping[1]
XFilterEvent[1]	XSetWMClientMachine[1]	XkbResizeKeyActions[1]
XFindContext[1]	XSetWMColormapWindows[1]	XkbResizeKeySyms[1]
XFindOnExtensionList[1]	XSetWMHints[1]	XkbResizeKeyType[1]
XFlush[1]	XSetWMIconName[1]	XkbSelectEventDetails[1]
XFlushGC[1]	XSetWMName[1]	XkbSelectEvents[1]
XFontsOfFontSet[1]	XSetWMNormalHints[1]	XkbSetAtomFunes[1]
XForceScreenSaver[1]	XSetWMProperties[1]	XkbSetAutoRepeatRate[1]
XFree[1]	XSetWMProtocols[1]	XkbSetAutoResetControls[1]
XFreeColormap[1]	XSetWMSizeHints[1]	XkbSetCompatMap[1]
XFreeColors[1]	XSetWindowBackground[1]	XkbSetControls[1]
XFreeCursor[1]	XSetWindowBackgroundPixmap[1]	XkbSetDebuggingFlags[1]
XFreeExtensionList[1]	XSetWindowBorder[1]	XkbSetDetectableAutoRepeat[1]
XFreeFont[1]	XSetWindowBorderPixmap[1]	XkbSetGeometry[1]
XFreeFontInfo[1]	XSetWindowBorderWidth[1]	XkbSetIgnoreLockMods[1]
XFreeFontNames[1]	XSetWindowColormap[1]	XkbSetIndicatorMap[1]

XFreeFontPath[1]	XSetZoomHints[1]	XkbSetMap[1]
XFreeFontSet[1]	XShrinkRegion[1]	XkbSetNamedIndicator[1]
XFreeGC[1]	XStoreBuffer[1]	XkbSetNames[1]
XFreeModifiermap[1]	XStoreBytes[1]	XkbSetServerInternalMods[1]
XFreePixmap[1]	XStoreColor[1]	XkbSetXlibControls[1]
XFreeStringList[1]	XStoreColors[1]	XkbToControl[1]
XGContextFromGC[1]	XStoreName[1]	XkbTranslateKeyCode[1]
XGeometry[1]	XStoreNamedColor[1]	XkbTranslateKeySym[1]
XGetAtomName[1]	XStringListToTextProperty[1]	XkbUpdateMapFromCore[1]
XGetAtomNames[1]	XStringToKeysym[1]	XkbUseExtension[1]
XGetClassHint[1]	XSubImage[1]	XkbVirtualModsToReal[1]
XGetCommand[1]	XSubtractRegion[1]	XmbDrawImageString[1]
XGetDefault[1]	XSupportsLocale[1]	XmbDrawString[1]
XGetErrorDatabaseText[1]	XSyne[1]	XmbDrawText[1]
XGetErrorText[1]	XTextExtents[1]	XmbLookupString[1]
XGetFontPath[1]	XTextExtents16[1]	XmbResetIC[1]
XGetFontProperty[1]	XTextPropertyToStringList[1]	XmbSetWMProperties[1]
XGetGCValues[1]	XTextWidth[1]	XmbTextEscapement[1]
XGetGeometry[1]	XTextWidth16[1]	XmbTextExtents[1]
XGetICValues[1]	XTranslateCoordinates[1]	XmbTextListToTextProperty[1]
XGetIMValues[1]	XUndefineCursor[1]	XmbTextPerCharExtents[1]
XGetIconName[1]	XUngrabButton[1]	XmbTextPropertyToTextList[1]
XGetIconSizes[1]	XUngrabKey[1]	XrmCombineDatabase[1]
XGetImage[1]	XUngrabKeyboard[1]	XrmCombineFileDatabase[1]
XGetInputFocus[1]	XUngrabPointer[1]	XrmDestroyDatabase[1]
XGetKeyboardControl[1]	XUngrabServer[1]	XrmEnumerateDatabase[1]
XGetKeyboardMapping[1]	XUninstallColormap[1]	XrmGetDatabase[1]
XGetModifierMapping[1]	XUnionRectWithRegion[1]	XrmGetFileDatabase[1]
XGetMotionEvents[1]	XUnionRegion[1]	XrmGetResource[1]
XGetNormalHints[1]	XUnloadFont[1]	XrmGetStringDatabase[1]
XGetOCValues[1]	XUnlockDisplay[1]	XrmInitialize[1]

XGetOMValues[1]	XUnmapSubwindows[1]	XrmLocaleOfDatabase[1]
XGetPixel[1]	XUnmapWindow[1]	XrmMergeDatabases[1]
XGetPointerControl[1]	XUnregisterIMInstantiateCallback[1]	XrmParseCommand[1]
XGetPointerMapping[1]	XUnsetICFocus[1]	XrmPermStringToQuark[1]
XGetRGBColormaps[1]	XVaCreateNestedList[1]	XrmPutFileDatabase[1]
XGetScreenSaver[1]	XVendorRelease[1]	XrmPutLineResource[1]
XGetSelectionOwner[1]	XVisualIDFromVisual[1]	XrmPutResource[1]
XGetSizeHints[1]	XWMGeometry[1]	XrmPutStringResource[1]
XGetStandardColormap[1]	XWarpPointer[1]	XrmQGetResource[1]
XGetSubImage[1]	XWhitePixel[1]	XrmQGetSearchList[1]
XGetTextProperty[1]	XWhitePixelOfScreen[1]	XrmQGetSearchResource[1]
XGetTransientForHint[1]	XWidthMMOfScreen[1]	XrmQPutResource[1]
XGetVisualInfo[1]	XWidthOfScreen[1]	XrmQPutStringResource[1]
XGetWMClientMachine[1]	XWindowEvent[1]	XrmQuarkToString[1]
XGetWMColormapWindows[1]	XWithdrawWindow[1]	XrmSetDatabase[1]
XGetWMHints[1]	XWriteBitmapFile[1]	XrmStringToBindingQuarkList[1]
XGetWMIconName[1]	XXorRegion[1]	XrmStringToQuark[1]
XGetWMName[1]	XauDisposeAuth[1]	XrmStringToQuarkList[1]
XGetWMNormalHints[1]	XauFileName[1]	XrmUniqueQuark[1]
XGetWMProtocols[1]	XauGetBestAuthByAddr[1]	Xutf8TextListToTextProperty[1]
XGetWMSizeHints[1]	XauReadAuth[1]	Xutf8TextPropertyToTextList[1]
XGetWindowAttributes[1]	XcmsAddColorSpace[1]	XweDrawImageString[1]
XGetWindowProperty[1]	XcmsAddFunctionSet[1]	XweDrawString[1]
XGetZoomHints[1]	XcmsAllocColor[1]	XweDrawText[1]
XGrabButton[1]	XcmsAllocNamedColor[1]	XweFreeStringList[1]
XGrabKey[1]	XcmsCCCOOfColormap[1]	XweLookupString[1]
XGrabKeyboard[1]	XcmsCIELabClipL[1]	XweResetIC[1]
XGrabPointer[1]	XcmsCIELabClipLab[1]	XweTextEscapement[1]
XGrabServer[1]	XcmsCIELabClipab[1]	XweTextExtents[1]
XHeightMMOfScreen[1]	XcmsCIELabQueryMaxC[1]	XweTextListToTextProperty[1]

XHeightOfScreen[1]	XcmsCIELabQueryMaxL[1]	XwcTextPerCharExtents[1]
XIMOfIC[1]	XcmsCIELabQueryMaxLC[1]	XwcTextPropertyToTextList[1]

SUSv2
ISO POSIX (2003)
SVID Issue 3
SVID Issue 4

Table A-2. libX11 Data 1. libc Function Interfaces

XSetAfterFunction_Exit(GLIBC_2.1.1)[1]	XSynchronizegetrlimit(GLIBC_2.1.1)[1]	sigandset(GLIBC_2.1.1)[1]
--	---------------------------------------	---------------------------

A.2. libXt

The behaviour of the interfaces in this library is specified by the following Standards.

Linux Standard Base

_IO_feof(GLIBC_2.0)[1]	getrlimit64(GLIBC_2.0)[1]	sigblock(GLIBC_2.0)[1]
_IO_getc(GLIBC_2.0)[1]	getrusage(GLIBC_2.0)[1]	sigdelset(GLIBC_2.0)[1]
_IO_putc(GLIBC_2.0)[1]	getservbyname(GLIBC_2.0)[1]	sigemptyset(GLIBC_2.0)[1]
_IO_puts(GLIBC_2.0)[1]	getservbyport(GLIBC_2.0)[1]	sigfillset(GLIBC_2.0)[1]
__assert_fail(GLIBC_2.0)[1]	getservent(GLIBC_2.0)[1]	siggetmask(GLIBC_2.0)[1]
__ctype_b_loc[1]	getsid()[1]	sighold()[1]
__ctype_get_mb_cur_max(GLIBC_2.0)[1]	getsockname(GLIBC_2.0)[1]	sigignore(GLIBC_2.0)[1]
__ctype_tolower_loc[1]	getsockopt()[1]	siginterrupt()[1]
__ctype_toupper_loc[1]	getsubopt()[1]	sigisemptyset()[1]
__cxa_atexit(GLIBC_2.1.3)[1]	gettext(GLIBC_2.1.3)[1]	sigismember(GLIBC_2.1.3)[1]
__errno_location(GLIBC_2.0)[1]	gettimeofday(GLIBC_2.0)[1]	siglongjmp(GLIBC_2.0)[1]
__fpending(GLIBC_2.2)[1]	getuid(GLIBC_2.2)[1]	signal(GLIBC_2.2)[1]
__fxstat(GLIBC_2.0)[1]	getutent(GLIBC_2.0)[1]	sigorset(GLIBC_2.0)[1]
__fxstat64(GLIBC_2.2)[1]	getutent_r(GLIBC_2.2)[1]	sigpause(GLIBC_2.2)[1]
__getpagesize(GLIBC_2.0)[1]	getutxent(GLIBC_2.0)[1]	sigpending(GLIBC_2.0)[1]
__getpgid(GLIBC_2.0)[1]	getutxid(GLIBC_2.0)[1]	sigprocmask(GLIBC_2.0)[1]
__h_errno_location[1]	getutxline()[1]	sigqueue()[1]
__isinf[1]	getw()[1]	sigrelse()[1]

__isinf[1]	getwc()[1]	sigreturn()[1]
__isinfl[1]	getwchar()[1]	sigset()[1]
__isnan[1]	getwd()[1]	sigstack()[1]
__isnanf[1]	glob()[1]	sigsuspend()[1]
__isnanl[1]	glob64()[1]	sigtimedwait()[1]
__libc_current_sigrtmax(GLIBC_2.1)[1]	globfree(GLIBC_2.1)[1]	sigwait(GLIBC_2.1)[1]
__libc_current_sigrtmin(GLIBC_2.1)[1]	globfree64(GLIBC_2.1)[1]	sigwaitinfo(GLIBC_2.1)[1]
__libc_start_main(GLIBC_2.0)[1]	gmtime(GLIBC_2.0)[1]	sleep(GLIBC_2.0)[1]
__lxstat(GLIBC_2.0)[1]	gmtime_r(GLIBC_2.0)[1]	snprintf(GLIBC_2.0)[1]
__lxstat64(GLIBC_2.2)[1]	grantpt(GLIBC_2.2)[1]	socket(GLIBC_2.2)[1]
__memcpy(GLIBC_2.0)[1]	hcreate(GLIBC_2.0)[1]	socketpair(GLIBC_2.0)[1]
__rawmemchr(GLIBC_2.1)[1]	hdestroy(GLIBC_2.1)[1]	sprintf(GLIBC_2.1)[1]
__register_atfork[1]	hsearch()[1]	srand()[1]
__sigsetjmp(GLIBC_2.0)[1]	htonl(GLIBC_2.0)[1]	srand48(GLIBC_2.0)[1]
__stpcpy(GLIBC_2.0)[1]	htons(GLIBC_2.0)[1]	srandom(GLIBC_2.0)[1]
__strdup(GLIBC_2.0)[1]	iconv(GLIBC_2.0)[1]	sscanf(GLIBC_2.0)[1]
__strtod_internal(GLIBC_2.0)[1]	iconv_close(GLIBC_2.0)[1]	statvfs(GLIBC_2.0)[1]
__strtof_internal(GLIBC_2.0)[1]	iconv_open(GLIBC_2.0)[1]	statvfs64[1]
__strtok_r(GLIBC_2.0)[1]	imaxabs(GLIBC_2.0)[1]	stime(GLIBC_2.0)[1]
__strtol_internal(GLIBC_2.0)[1]	imaxdiv(GLIBC_2.0)[1]	stpcpy(GLIBC_2.0)[1]
__strtold_internal(GLIBC_2.0)[1]	index(GLIBC_2.0)[1]	stpncpy(GLIBC_2.0)[1]
__strtoll_internal(GLIBC_2.0)[1]	inet_addr(GLIBC_2.0)[1]	strcasecmp(GLIBC_2.0)[1]
__strtoul_internal(GLIBC_2.0)[1]	inet_ntoa(GLIBC_2.0)[1]	strcasestr(GLIBC_2.0)[1]
__strtoull_internal(GLIBC_2.0)[1]	inet_ntop[1]	strcat(GLIBC_2.0)[1]
__sysconf(GLIBC_2.2)[1]	inet_pton[1]	strchr(GLIBC_2.2)[1]
__sysv_signal(GLIBC_2.0)[1]	initgroups(GLIBC_2.0)[1]	strcmp(GLIBC_2.0)[1]
__wcstod_internal(GLIBC_2.0)[1]	initstate(GLIBC_2.0)[1]	strcoll(GLIBC_2.0)[1]
__wcstof_internal(GLIBC_2.0)[1]	insque(GLIBC_2.0)[1]	strcpy(GLIBC_2.0)[1]
__westol_internal(GLIBC_2.0)[1]	ioctl(GLIBC_2.0)[1]	strcspn(GLIBC_2.0)[1]

__wcstold_internal(GLIBC_2.0)[1]	isalnum(GLIBC_2.0)[1]	strdup(GLIBC_2.0)[1]
__wcstoul_internal(GLIBC_2.0)[1]	isalpha(GLIBC_2.0)[1]	strerror(GLIBC_2.0)[1]
__xmknod(GLIBC_2.0)[1]	isascii(GLIBC_2.0)[1]	strerror_r(GLIBC_2.0)[1]
__xstat(GLIBC_2.0)[1]	isatty(GLIBC_2.0)[1]	strfmon(GLIBC_2.0)[1]
__xstat64(GLIBC_2.2)[1]	isblank(GLIBC_2.2)[1]	strfry(GLIBC_2.2)[1]
_exit(GLIBC_2.0)[1]	iscntrl(GLIBC_2.0)[1]	strftime(GLIBC_2.0)[1]
_longjmp(GLIBC_2.0)[1]	isdigit(GLIBC_2.0)[1]	strlen(GLIBC_2.0)[1]
_obstack_begin(GLIBC_2.0)[1]	isgraph(GLIBC_2.0)[1]	strncasecmp(GLIBC_2.0)[1]
_obstack_newchunk(GLIBC_2.0)[1]	islower(GLIBC_2.0)[1]	strncat(GLIBC_2.0)[1]
_setjmp(GLIBC_2.0)[1]	isprint(GLIBC_2.0)[1]	strncmp(GLIBC_2.0)[1]
_tolower(GLIBC_2.0)[1]	ispunct(GLIBC_2.0)[1]	strncpy(GLIBC_2.0)[1]
_toupper(GLIBC_2.0)[1]	isspace(GLIBC_2.0)[1]	strndup(GLIBC_2.0)[1]
a64l(GLIBC_2.0)[1]	isupper(GLIBC_2.0)[1]	strnlen(GLIBC_2.0)[1]
abort(GLIBC_2.0)[1]	iswalnum(GLIBC_2.0)[1]	strpbrk(GLIBC_2.0)[1]
abs(GLIBC_2.0)[1]	iswalpha(GLIBC_2.0)[1]	strptime(GLIBC_2.0)[1]
accept(GLIBC_2.0)[1]	iswblank(GLIBC_2.0)[1]	strrchr(GLIBC_2.0)[1]
access(GLIBC_2.0)[1]	iswcntrl(GLIBC_2.0)[1]	strsep(GLIBC_2.0)[1]
acct(GLIBC_2.0)[1]	iswctype(GLIBC_2.0)[1]	strsignal(GLIBC_2.0)[1]
adjtime(GLIBC_2.0)[1]	iswdigit(GLIBC_2.0)[1]	strspn(GLIBC_2.0)[1]
alarm(GLIBC_2.0)[1]	iswgraph(GLIBC_2.0)[1]	strstr(GLIBC_2.0)[1]
asctime(GLIBC_2.0)[1]	iswlower(GLIBC_2.0)[1]	strtod(GLIBC_2.0)[1]
asctime_r(GLIBC_2.0)[1]	iswprint(GLIBC_2.0)[1]	strtof(GLIBC_2.0)[1]
asprintf(GLIBC_2.0)[1]	iswpunct(GLIBC_2.0)[1]	strtoimax(GLIBC_2.0)[1]
atof(GLIBC_2.0)[1]	iswspace(GLIBC_2.0)[1]	strtok(GLIBC_2.0)[1]
atoi(GLIBC_2.0)[1]	iswupper(GLIBC_2.0)[1]	strtok_r(GLIBC_2.0)[1]
atol(GLIBC_2.0)[1]	iswxdigit(GLIBC_2.0)[1]	strtol(GLIBC_2.0)[1]
atoll[1]	isxdigit[1]	strtold[1]
authnone_create(GLIBC_2.0)[1]	jrand48(GLIBC_2.0)[1]	strtoll(GLIBC_2.0)[1]
basename(GLIBC_2.0)[1]	key_decryptsession(GLIBC_2.0)[1]	strtoq(GLIBC_2.0)[1]
bcmp(GLIBC_2.0)[1]	kill(GLIBC_2.0)[1]	strtoul(GLIBC_2.0)[1]

bcopy(GLIBC_2.0)[1]	killpg(GLIBC_2.0)[1]	strtoull(GLIBC_2.0)[1]
bind(GLIBC_2.0)[1]	l64a(GLIBC_2.0)[1]	strtoumax(GLIBC_2.0)[1]
bind_textdomain_codeset[1]	labs()[1]	strtouq()[1]
bindresvport(GLIBC_2.0)[1]	lchown(GLIBC_2.0)[1]	strverscmp(GLIBC_2.0)[1]
bindtextdomain(GLIBC_2.0)[1]	lcong48(GLIBC_2.0)[1]	strxfrm(GLIBC_2.0)[1]
brk(GLIBC_2.0)[1]	ldiv(GLIBC_2.0)[1]	svc_getreqset(GLIBC_2.0)[1]
bsd_signal(GLIBC_2.0)[1]	lfind(GLIBC_2.0)[1]	svc_register(GLIBC_2.0)[1]
bsearch(GLIBC_2.0)[1]	link(GLIBC_2.0)[1]	svc_run(GLIBC_2.0)[1]
btowc(GLIBC_2.0)[1]	listen(GLIBC_2.0)[1]	svc_sendreply(GLIBC_2.0)[1]
bzero(GLIBC_2.0)[1]	llabs(GLIBC_2.0)[1]	svcerr_auth(GLIBC_2.0)[1]
calloc(GLIBC_2.0)[1]	lldiv(GLIBC_2.0)[1]	svcerr_decode(GLIBC_2.0)[1]
catclose(GLIBC_2.0)[1]	localeconv(GLIBC_2.0)[1]	svcerr_noproc(GLIBC_2.0)[1]
catgets(GLIBC_2.0)[1]	localtime(GLIBC_2.0)[1]	svcerr_noprog(GLIBC_2.0)[1]
catopen(GLIBC_2.0)[1]	localtime_r(GLIBC_2.0)[1]	svcerr_progvers(GLIBC_2.0)[1]
cfgetispeed(GLIBC_2.0)[1]	lockf(GLIBC_2.0)[1]	svcerr_systemerr(GLIBC_2.0)[1]
cfgetospeed(GLIBC_2.0)[1]	lockf64(GLIBC_2.0)[1]	svcerr_weakauth(GLIBC_2.0)[1]
cfmakeraw(GLIBC_2.0)[1]	longjmp(GLIBC_2.0)[1]	svctcp_create(GLIBC_2.0)[1]
cfsetispeed(GLIBC_2.0)[1]	lrand48(GLIBC_2.0)[1]	svcudp_create(GLIBC_2.0)[1]
cfsetospeed(GLIBC_2.0)[1]	lsearch(GLIBC_2.0)[1]	swab(GLIBC_2.0)[1]
cfsetspeed(GLIBC_2.0)[1]	lseek(GLIBC_2.0)[1]	swapcontext(GLIBC_2.0)[1]
chdir(GLIBC_2.0)[1]	lseek64(GLIBC_2.0)[1]	swprintf(GLIBC_2.0)[1]
chmod(GLIBC_2.0)[1]	makecontext(GLIBC_2.0)[1]	swscanf(GLIBC_2.0)[1]
chown(GLIBC_2.1)[1]	malloc(GLIBC_2.1)[1]	symlink(GLIBC_2.1)[1]
chroot(GLIBC_2.0)[1]	mblen(GLIBC_2.0)[1]	sync(GLIBC_2.0)[1]
clearerr(GLIBC_2.0)[1]	mbrlen(GLIBC_2.0)[1]	sysconf(GLIBC_2.0)[1]
clnt_create(GLIBC_2.0)[1]	mbrtowc(GLIBC_2.0)[1]	syslog(GLIBC_2.0)[1]
clnt_pcreateerror(GLIBC_2.0)[1]	mbsinit(GLIBC_2.0)[1]	system(GLIBC_2.0)[1]
clnt_perrno(GLIBC_2.0)[1]	mbsnrtowcs(GLIBC_2.0)[1]	tcdrain(GLIBC_2.0)[1]
clnt_perror(GLIBC_2.0)[1]	mbsrtowcs(GLIBC_2.0)[1]	tcflow(GLIBC_2.0)[1]
clnt_spcreateerror(GLIBC_2.0)[1]	mbstowcs(GLIBC_2.0)[1]	tcflush(GLIBC_2.0)[1]
clnt_sperrno(GLIBC_2.0)[1]	mbtowc(GLIBC_2.0)[1]	tcgetattr(GLIBC_2.0)[1]

clnt_spperror(GLIBC_2.0)[1]	memccpy(GLIBC_2.0)[1]	tcgetpgrp(GLIBC_2.0)[1]
clock(GLIBC_2.0)[1]	memchr(GLIBC_2.0)[1]	tcgetsid(GLIBC_2.0)[1]
close(GLIBC_2.0)[1]	memcmp(GLIBC_2.0)[1]	tcsendbreak(GLIBC_2.0)[1]
closedir(GLIBC_2.0)[1]	memcpy(GLIBC_2.0)[1]	tcsetattr(GLIBC_2.0)[1]
closelog(GLIBC_2.0)[1]	memmem(GLIBC_2.0)[1]	tcsetpgrp(GLIBC_2.0)[1]
confstr(GLIBC_2.0)[1]	memmove(GLIBC_2.0)[1]	tdelete[1]
connect(GLIBC_2.0)[1]	memrchr(GLIBC_2.0)[1]	telldir(GLIBC_2.0)[1]
creat(GLIBC_2.0)[1]	memset(GLIBC_2.0)[1]	tempnam(GLIBC_2.0)[1]
creat64(GLIBC_2.1)[1]	mkdir(GLIBC_2.1)[1]	textdomain(GLIBC_2.1)[1]
ctermid(GLIBC_2.0)[1]	mkfifo(GLIBC_2.0)[1]	tfind(GLIBC_2.0)[1]
ctime(GLIBC_2.0)[1]	mkstemp(GLIBC_2.0)[1]	time(GLIBC_2.0)[1]
ctime_r(GLIBC_2.0)[1]	mkstemp64(GLIBC_2.0)[1]	times(GLIBC_2.0)[1]
cuserid(GLIBC_2.0)[1]	mktemp(GLIBC_2.0)[1]	tmpfile(GLIBC_2.0)[1]
daemon(GLIBC_2.0)[1]	mktime(GLIBC_2.0)[1]	tmpfile64(GLIBC_2.0)[1]
dcgettext(GLIBC_2.0)[1]	mlock(GLIBC_2.0)[1]	tmpnam(GLIBC_2.0)[1]
dcngettext[1]	mlockall()[1]	toascii()[1]
dgettext[1]	mmap()[1]	tolower()[1]
difftime(GLIBC_2.0)[1]	mmap64(GLIBC_2.0)[1]	toupper(GLIBC_2.0)[1]
dirname(GLIBC_2.0)[1]	mprotect(GLIBC_2.0)[1]	towctrans(GLIBC_2.0)[1]
div(GLIBC_2.0)[1]	mrand48(GLIBC_2.0)[1]	tolower(GLIBC_2.0)[1]
dngettext[1]	msgctl()[1]	towupper()[1]
drand48(GLIBC_2.0)[1]	msgget(GLIBC_2.0)[1]	truncate(GLIBC_2.0)[1]
dup(GLIBC_2.0)[1]	msgrcv(GLIBC_2.0)[1]	truncate64(GLIBC_2.0)[1]
dup2(GLIBC_2.0)[1]	msgsnd(GLIBC_2.0)[1]	tsearch(GLIBC_2.0)[1]
ecvt(GLIBC_2.0)[1]	msync(GLIBC_2.0)[1]	ttynam(GLIBC_2.0)[1]
endgrent(GLIBC_2.0)[1]	munlock(GLIBC_2.0)[1]	ttynam_r(GLIBC_2.0)[1]
endnetent(GLIBC_2.0)[1]	munlockall(GLIBC_2.0)[1]	twalk(GLIBC_2.0)[1]
endprotoent(GLIBC_2.0)[1]	munmap(GLIBC_2.0)[1]	tzset(GLIBC_2.0)[1]
endpwent(GLIBC_2.0)[1]	nanosleep(GLIBC_2.0)[1]	ualarm(GLIBC_2.0)[1]
endservent(GLIBC_2.0)[1]	nftw(GLIBC_2.0)[1]	ulimit(GLIBC_2.0)[1]
endutent(GLIBC_2.0)[1]	nftw64(GLIBC_2.0)[1]	umask(GLIBC_2.0)[1]

endutxent(GLIBC_2.1)[1]	ngettext[1]	uname(GLIBC_2.1)[1]
erand48(GLIBC_2.0)[1]	nice(GLIBC_2.0)[1]	ungetc(GLIBC_2.0)[1]
err(GLIBC_2.0)[1]	nl_langinfo(GLIBC_2.0)[1]	ungetwc(GLIBC_2.0)[1]
error(GLIBC_2.0)[1]	nrnd48(GLIBC_2.0)[1]	unlink(GLIBC_2.0)[1]
errx(GLIBC_2.0)[1]	ntohl(GLIBC_2.0)[1]	unlockpt(GLIBC_2.0)[1]
execl(GLIBC_2.0)[1]	ntohs(GLIBC_2.0)[1]	unsetenv[1]
execle(GLIBC_2.0)[1]	obstack_free(GLIBC_2.0)[1]	usleep(GLIBC_2.0)[1]
execlp(GLIBC_2.0)[1]	open(GLIBC_2.0)[1]	utime(GLIBC_2.0)[1]
execv(GLIBC_2.0)[1]	open64(GLIBC_2.0)[1]	utimes(GLIBC_2.0)[1]
execve(GLIBC_2.0)[1]	opendir(GLIBC_2.0)[1]	vasprintf(GLIBC_2.0)[1]
execvp(GLIBC_2.0)[1]	openlog(GLIBC_2.0)[1]	vdprintf(GLIBC_2.0)[1]
exit(GLIBC_2.0)[1]	pathconf(GLIBC_2.0)[1]	verrx(GLIBC_2.0)[1]
fchdir(GLIBC_2.0)[1]	pause(GLIBC_2.0)[1]	vfork(GLIBC_2.0)[1]
fchmod(GLIBC_2.0)[1]	pclose(GLIBC_2.0)[1]	vfprintf(GLIBC_2.0)[1]
fchown(GLIBC_2.0)[1]	perror(GLIBC_2.0)[1]	vfscanf[1]
fclose(GLIBC_2.1)[1]	pipe(GLIBC_2.1)[1]	vfwprintf(GLIBC_2.1)[1]
fcntl(GLIBC_2.0)[1]	pmap_getport(GLIBC_2.0)[1]	vfwscanf(GLIBC_2.0)[1]
fcvt(GLIBC_2.0)[1]	pmap_set(GLIBC_2.0)[1]	vprintf(GLIBC_2.0)[1]
fdatasync(GLIBC_2.0)[1]	pmap_unset(GLIBC_2.0)[1]	vscanf[1]
fdopen(GLIBC_2.1)[1]	poll(GLIBC_2.1)[1]	vsprintf(GLIBC_2.1)[1]
feof(GLIBC_2.0)[1]	popen(GLIBC_2.0)[1]	vsprintf(GLIBC_2.0)[1]
ferror(GLIBC_2.0)[1]	posix_memalign(GLIBC_2.0)[1]	vsscanf[1]
fflush(GLIBC_2.0)[1]	printf(GLIBC_2.0)[1]	vswprintf(GLIBC_2.0)[1]
fflush_unlocked(GLIBC_2.0)[1]	psignal(GLIBC_2.0)[1]	vswscanf(GLIBC_2.0)[1]
ffs(GLIBC_2.0)[1]	ptsname(GLIBC_2.0)[1]	vsyslog[1]
fgetc(GLIBC_2.0)[1]	putc(GLIBC_2.0)[1]	vwprintf(GLIBC_2.0)[1]
fgetpos(GLIBC_2.0)[1]	putc_unlocked(GLIBC_2.0)[1]	vwscanf(GLIBC_2.0)[1]
fgetpos64(GLIBC_2.1)[1]	putchar(GLIBC_2.1)[1]	wait(GLIBC_2.1)[1]
fgets(GLIBC_2.0)[1]	putchar_unlocked(GLIBC_2.0)[1]	wait3(GLIBC_2.0)[1]
fgetwc(GLIBC_2.2)[1]	putenv(GLIBC_2.2)[1]	wait4(GLIBC_2.2)[1]
fgetwc_unlocked(GLIBC_2.2)[1]	puts(GLIBC_2.2)[1]	waitpid(GLIBC_2.2)[1]

fgetws(GLIBC_2.2)[1]	pututxline(GLIBC_2.2)[1]	warn(GLIBC_2.2)[1]
fileno(GLIBC_2.0)[1]	putw(GLIBC_2.0)[1]	warnx(GLIBC_2.0)[1]
flock(GLIBC_2.0)[1]	putwc(GLIBC_2.0)[1]	wcpcpy(GLIBC_2.0)[1]
flockfile(GLIBC_2.0)[1]	putwchar(GLIBC_2.0)[1]	wcpncpy(GLIBC_2.0)[1]
fmtmsg(GLIBC_2.1)[1]	qsort(GLIBC_2.1)[1]	wcrtomb(GLIBC_2.1)[1]
fnmatch(GLIBC_2.2.3)[1]	raise(GLIBC_2.2.3)[1]	wscasecmp(GLIBC_2.2.3)[1]
fopen(GLIBC_2.1)[1]	rand(GLIBC_2.1)[1]	wscat(GLIBC_2.1)[1]
fopen64(GLIBC_2.1)[1]	rand_r(GLIBC_2.1)[1]	wcschr(GLIBC_2.1)[1]
fork(GLIBC_2.0)[1]	random(GLIBC_2.0)[1]	wscmp(GLIBC_2.0)[1]
fpathconf(GLIBC_2.0)[1]	random_r(GLIBC_2.0)[1]	wscoll(GLIBC_2.0)[1]
fprintf(GLIBC_2.0)[1]	read(GLIBC_2.0)[1]	wscpy(GLIBC_2.0)[1]
fputc(GLIBC_2.0)[1]	readdir(GLIBC_2.0)[1]	wscspn(GLIBC_2.0)[1]
fputs(GLIBC_2.0)[1]	readdir64(GLIBC_2.0)[1]	wcsdup(GLIBC_2.0)[1]
fputwc(GLIBC_2.2)[1]	readdir_r[1]	wcsftime(GLIBC_2.2)[1]
fputws(GLIBC_2.2)[1]	readlink(GLIBC_2.2)[1]	wcslen(GLIBC_2.2)[1]
fread(GLIBC_2.0)[1]	readv(GLIBC_2.0)[1]	wcsncasecmp(GLIBC_2.0)[1]
free(GLIBC_2.0)[1]	realloc(GLIBC_2.0)[1]	wcsncat(GLIBC_2.0)[1]
freeaddrinfo[1]	realpath()[1]	wcsncmp()[1]
freopen(GLIBC_2.0)[1]	recv(GLIBC_2.0)[1]	wcsncpy(GLIBC_2.0)[1]
freopen64(GLIBC_2.1)[1]	recvfrom(GLIBC_2.1)[1]	wcsnlen(GLIBC_2.1)[1]
fscanf(GLIBC_2.0)[1]	recvmsg(GLIBC_2.0)[1]	wcsnrtombs(GLIBC_2.0)[1]
fseek(GLIBC_2.0)[1]	regcomp(GLIBC_2.0)[1]	wcspbrk(GLIBC_2.0)[1]
fseeko(GLIBC_2.1)[1]	regerror(GLIBC_2.1)[1]	wcsrchr(GLIBC_2.1)[1]
fseeko64(GLIBC_2.1)[1]	regexexec(GLIBC_2.1)[1]	wcsrtombs(GLIBC_2.1)[1]
fsetpos(GLIBC_2.0)[1]	regfree(GLIBC_2.0)[1]	wcsspn(GLIBC_2.0)[1]
fsetpos64(GLIBC_2.1)[1]	remove(GLIBC_2.1)[1]	wcsstr(GLIBC_2.1)[1]
fstatvfs(GLIBC_2.1)[1]	remque(GLIBC_2.1)[1]	wcstod(GLIBC_2.1)[1]
fstatvfs64(GLIBC_2.1)[1]	rename(GLIBC_2.1)[1]	wcstof(GLIBC_2.1)[1]
fsync(GLIBC_2.0)[1]	rewind(GLIBC_2.0)[1]	wcstoimax(GLIBC_2.0)[1]
ftell(GLIBC_2.0)[1]	rewinddir(GLIBC_2.0)[1]	wcstok(GLIBC_2.0)[1]
ftello(GLIBC_2.1)[1]	rindex(GLIBC_2.1)[1]	wcstol(GLIBC_2.1)[1]

ftello64(GLIBC_2.1)[1]	rmdir(GLIBC_2.1)[1]	wcstold(GLIBC_2.1)[1]
ftime(GLIBC_2.0)[1]	sbrk(GLIBC_2.0)[1]	wcstoll(GLIBC_2.0)[1]
ftok(GLIBC_2.0)[1]	scanf(GLIBC_2.0)[1]	wcstombs(GLIBC_2.0)[1]
ftruncate(GLIBC_2.0)[1]	sched_get_priority_max(GLIBC_2.0)[1]	wcstoq(GLIBC_2.0)[1]
ftruncate64(GLIBC_2.1)[1]	sched_get_priority_min(GLIBC_2.1)[1]	wcstoul(GLIBC_2.1)[1]
ftrylockfile(GLIBC_2.0)[1]	sched_getparam(GLIBC_2.0)[1]	wcstoull(GLIBC_2.0)[1]
ftw(GLIBC_2.0)[1]	sched_getscheduler(GLIBC_2.0)[1]	wcstoumax(GLIBC_2.0)[1]
ftw64(GLIBC_2.1)[1]	sched_rr_get_interval(GLIBC_2.1)[1]	wcstouq(GLIBC_2.1)[1]
funlockfile(GLIBC_2.0)[1]	sched_setparam(GLIBC_2.0)[1]	wcswcs(GLIBC_2.0)[1]
fwide(GLIBC_2.2)[1]	sched_setscheduler(GLIBC_2.2)[1]	wcswidth(GLIBC_2.2)[1]
fwprintf(GLIBC_2.2)[1]	sched_yield(GLIBC_2.2)[1]	wcsxfrm(GLIBC_2.2)[1]
fwrite(GLIBC_2.0)[1]	seed48(GLIBC_2.0)[1]	wctob(GLIBC_2.0)[1]
fwscanf(GLIBC_2.2)[1]	seekdir(GLIBC_2.2)[1]	wctomb(GLIBC_2.2)[1]
gai_strerror[1]	select()[1]	wctrans()[1]
gcvrt(GLIBC_2.0)[1]	semctl(GLIBC_2.0)[1]	wctype(GLIBC_2.0)[1]
getaddrinfo[1]	semget()[1]	wcwidth()[1]
getc(GLIBC_2.0)[1]	semop(GLIBC_2.0)[1]	wmemchr(GLIBC_2.0)[1]
getc_unlocked(GLIBC_2.0)[1]	send(GLIBC_2.0)[1]	wmemcmp(GLIBC_2.0)[1]
getchar(GLIBC_2.0)[1]	sendmsg(GLIBC_2.0)[1]	wmemcpy(GLIBC_2.0)[1]
getchar_unlocked(GLIBC_2.0)[1]	sendto(GLIBC_2.0)[1]	wmemmove(GLIBC_2.0)[1]
getcontext(GLIBC_2.1)[1]	setbuf(GLIBC_2.1)[1]	wmemset(GLIBC_2.1)[1]
getcwd(GLIBC_2.0)[1]	setbuffer(GLIBC_2.0)[1]	wordexp(GLIBC_2.0)[1]
getdate(GLIBC_2.1)[1]	setcontext(GLIBC_2.1)[1]	wordfree(GLIBC_2.1)[1]
getdomainname(GLIBC_2.0)[1]	setdomainname[1]	wprintf(GLIBC_2.0)[1]
getegid(GLIBC_2.0)[1]	setegid(GLIBC_2.0)[1]	write(GLIBC_2.0)[1]
getenv(GLIBC_2.0)[1]	setenv[1]	writew(GLIBC_2.0)[1]
geteuid(GLIBC_2.0)[1]	seteuid(GLIBC_2.0)[1]	wscanf(GLIBC_2.0)[1]
getgid(GLIBC_2.0)[1]	setgid(GLIBC_2.0)[1]	xdr_accepted_reply(GLIBC_2.0)[1]
getgrent(GLIBC_2.0)[1]	setgrent(GLIBC_2.0)[1]	xdr_array(GLIBC_2.0)[1]

getgrgid(GLIBC_2.0)[1]	setgroups(GLIBC_2.0)[1]	xdr_bool(GLIBC_2.0)[1]
getgrgid_r(GLIBC_2.0)[1]	sethostid(GLIBC_2.0)[1]	xdr_bytes(GLIBC_2.0)[1]
getgrnam(GLIBC_2.0)[1]	sethostname(GLIBC_2.0)[1]	xdr_callhdr(GLIBC_2.0)[1]
getgrnam_r(GLIBC_2.0)[1]	setitimer(GLIBC_2.0)[1]	xdr_callmsg(GLIBC_2.0)[1]
getgroups(GLIBC_2.0)[1]	setlocale(GLIBC_2.0)[1]	xdr_char(GLIBC_2.0)[1]
gethostbyaddr(GLIBC_2.0)[1]	setlogmask(GLIBC_2.0)[1]	xdr_double(GLIBC_2.0)[1]
gethostbyname(GLIBC_2.0)[1]	setnetent(GLIBC_2.0)[1]	xdr_enum(GLIBC_2.0)[1]
gethostid(GLIBC_2.0)[1]	setpgid(GLIBC_2.0)[1]	xdr_float(GLIBC_2.0)[1]
gethostname(GLIBC_2.0)[1]	setpgrp(GLIBC_2.0)[1]	xdr_free(GLIBC_2.0)[1]
getitimer(GLIBC_2.0)[1]	setpriority(GLIBC_2.0)[1]	xdr_int(GLIBC_2.0)[1]
getloadavg(GLIBC_2.2)[1]	setprotoent(GLIBC_2.2)[1]	xdr_long(GLIBC_2.2)[1]
getlogin(GLIBC_2.0)[1]	setpwent(GLIBC_2.0)[1]	xdr_opaque(GLIBC_2.0)[1]
getnameinfo[1]	setregid()[1]	xdr_opaque_auth()[1]
getnetbyaddr(GLIBC_2.0)[1]	setreuid(GLIBC_2.0)[1]	xdr_pointer(GLIBC_2.0)[1]
getopt(GLIBC_2.0)[1]	setrlimit(GLIBC_2.0)[1]	xdr_reference(GLIBC_2.0)[1]
getopt_long(GLIBC_2.0)[1]	setrlimit64[1]	xdr_rejected_reply(GLIBC_2.0)[1]
getopt_long_only(GLIBC_2.0)[1]	setservent(GLIBC_2.0)[1]	xdr_replymsg(GLIBC_2.0)[1]
getpagesize(GLIBC_2.0)[1]	setsid(GLIBC_2.0)[1]	xdr_short(GLIBC_2.0)[1]
getpeername(GLIBC_2.0)[1]	setsockopt(GLIBC_2.0)[1]	xdr_string(GLIBC_2.0)[1]
getpgid(GLIBC_2.0)[1]	setstate(GLIBC_2.0)[1]	xdr_u_char(GLIBC_2.0)[1]
getpgrp(GLIBC_2.0)[1]	setuid(GLIBC_2.0)[1]	xdr_u_int(GLIBC_2.0)[1]
getpid(GLIBC_2.0)[1]	setutent(GLIBC_2.0)[1]	xdr_u_long(GLIBC_2.0)[1]
getppid(GLIBC_2.0)[1]	setutxent(GLIBC_2.0)[1]	xdr_u_short(GLIBC_2.0)[1]
getpriority(GLIBC_2.0)[1]	setvbuf(GLIBC_2.0)[1]	xdr_union(GLIBC_2.0)[1]
getprotobyname(GLIBC_2.0)[1]	shmat(GLIBC_2.0)[1]	xdr_vector(GLIBC_2.0)[1]
getprotobynumber(GLIBC_2.0)[1]	shmctl(GLIBC_2.0)[1]	xdr_void(GLIBC_2.0)[1]
getprotoent(GLIBC_2.0)[1]	shmdt(GLIBC_2.0)[1]	xdr_wrapstring(GLIBC_2.0)[1]
getpwent(GLIBC_2.0)[1]	shmget(GLIBC_2.0)[1]	xdrmem_create(GLIBC_2.0)[1]
getpwnam(GLIBC_2.0)[1]	shutdown(GLIBC_2.0)[1]	xdrrec_create(GLIBC_2.0)[1]
getpwnam_r(GLIBC_2.0)[1]	sigaction(GLIBC_2.0)[1]	xdrrec_eof(GLIBC_2.0)[1]
getpwuid(GLIBC_2.0)[1]	sigaddset(GLIBC_2.0)[1]	

getpwuid_r(GLIBC_2.0)[1]	sigaltstack(GLIBC_2.0)[1]	
--------------------------	---------------------------	--

Table A-3. libXt Function2. libc Data Interfaces

XtAddAccelerations[1]__daylightID STD 46 LSB	XtCvtStringToInitialState[1]__time zoneID STD 46 LSB	XtOwnSelectionIncremental[1]_sys_errlistID STD 46 LSB
XtAddCallback[1]__environID STD 46 LSB	XtCvtStringToInt[1]__tznameID STD 46 LSB	XtParent[1]
XtAddCallbacks[1]	XtCvtStringToPixel[1]	XtParseAcceleratorTable[1]

A.2. libcrypt

The behaviour of the interfaces in this library is specified by the following Standards.

ISO POSIX (2003)

Table A-3. libcrypt Function Interfaces

XtAddConverter[1]crypt(GLIBC_2.0)[1]	XtCvtStringToRestartStyle[1]encry pt(GLIBC_2.0)[1]	XtParseTranslationTable[1]setkey(GLIBC_2.0)[1]
XtAddEventHandler[1]	XtCvtStringToShort[1]	XtPeekEvent[1]
XtAddExposureToRegion[1]	XtCvtStringToTranslationTable[1]	XtPending[1]
XtAddGrab[1]	XtCvtStringToUnsignedChar[1]	XtPopdown[1]
XtAddInput[1]	XtCvtStringToVisual[1]	XtPopup[1]
XtAddRawEventHandler[1]	XtDatabase[1]	XtPopupSpringLoaded[1]
XtAddSignal[1]	XtDestroyApplicationContext[1]	XtProcessEvent[1]
XtAddTimeOut[1]	XtDestroyGC[1]	XtProcessLock[1]
XtAddWorkProc[1]	XtDestroyWidget[1]	XtProcessUnlock[1]
XtAllocateGC[1]	XtDirectConvert[1]	XtQueryGeometry[1]
XtAppAddActionHook[1]	XtDisownSelection[1]	XtRealizeWidget[1]
XtAppAddActions[1]	XtDispatchEvent[1]	XtRealloc[1]
XtAppAddBlockHook[1]	XtDispatchEventToWidget[1]	XtRegisterCaseConverter[1]
XtAppAddConverter[1]	XtDisplay[1]	XtRegisterDrawable[1]
XtAppAddInput[1]	XtDisplayInitialize[1]	XtRegisterExtensionSelector[1]
XtAppAddSignal[1]	XtDisplayOfObject[1]	XtRegisterGrabAction[1]
XtAppAddTimeOut[1]	XtDisplayStringConversionWarnin	XtReleaseGC[1]

	g[1]	
XtAppAddWorkProc[1]	XtDisplayToApplicationContext[1]	XtReleasePropertyAtom[1]
XtAppCreateShell[1]	XtError[1]	XtRemoveActionHook[1]
XtAppError[1]	XtErrorMsg[1]	XtRemoveAllCallbacks[1]
XtAppErrorMsg[1]	XtFindFile[1]	XtRemoveBlockHook[1]
XtAppGetErrorDatabase[1]	XtFree[1]	XtRemoveCallback[1]
XtAppGetErrorDatabaseText[1]	XtGetActionKeysym[1]	XtRemoveCallbacks[1]
XtAppGetExitFlag[1]	XtGetActionList[1]	XtRemoveEventHandler[1]
XtAppGetSelectionTimeout[1]	XtGetApplicationNameAndClass[1]	XtRemoveEventTypeHandler[1]
XtAppInitialize[1]	XtGetApplicationResources[1]	XtRemoveGrab[1]
XtAppLock[1]	XtGetClassExtension[1]	XtRemoveInput[1]
XtAppMainLoop[1]	XtGetConstraintResourceList[1]	XtRemoveRawEventHandler[1]
XtAppNextEvent[1]	XtGetDisplays[1]	XtRemoveSignal[1]
XtAppPeekEvent[1]	XtGetErrorDatabase[1]	XtRemoveTimeOut[1]
XtAppPending[1]	XtGetErrorDatabaseText[1]	XtRemoveWorkProc[1]
XtAppProcessEvent[1]	XtGetGC[1]	XtReservePropertyAtom[1]
XtAppReleaseCacheRefs[1]	XtGetKeyboardFocusWidget[1]	XtResizeWidget[1]
XtAppSetErrorHandler[1]	XtGetKeysymTable[1]	XtResizeWindow[1]
XtAppSetErrorMsgHandler[1]	XtGetMultiClickTime[1]	XtResolvePathname[1]
XtAppSetExitFlag[1]	XtGetResourceList[1]	XtScreen[1]
XtAppSetFallbackResources[1]	XtGetSelectionParameters[1]	XtScreenDatabase[1]
XtAppSetSelectionTimeout[1]	XtGetSelectionRequest[1]	XtScreenOfObject[1]
XtAppSetTypeConverter[1]	XtGetSelectionTimeout[1]	XtSendSelectionRequest[1]
XtAppSetWarningHandler[1]	XtGetSelectionValue[1]	XtSessionGetToken[1]
XtAppSetWarningMsgHandler[1]	XtGetSelectionValueIncremental[1]	XtSessionReturnToken[1]
XtAppUnlock[1]	XtGetSelectionValues[1]	XtSetErrorHandler[1]
XtAppWarning[1]	XtGetSelectionValuesIncremental[1]	XtSetErrorMsgHandler[1]
XtAppWarningMsg[1]	XtGetSubresources[1]	XtSetEventDispatcher[1]
XtAugmentTranslations[1]	XtGetSubvalues[1]	XtSetKeyTranslator[1]
XtBuildEventMask[1]	XtGetValues[1]	XtSetKeyboardFocus[1]

XtCallAcceptFocus[1]	XtGrabButton[1]	XtSetLanguageProc[1]
XtCallActionProc[1]	XtGrabKey[1]	XtSetMappedWhenManaged[1]
XtCallCallbackList[1]	XtGrabKeyboard[1]	XtSetMultiClickTime[1]
XtCallCallbacks[1]	XtGrabPointer[1]	XtSetSelectionParameters[1]
XtCallConverter[1]	XtHasCallbacks[1]	XtSetSelectionTimeout[1]
XtCallbackExclusive[1]	XtHooksOfDisplay[1]	XtSetSensitive[1]
XtCallbackNone[1]	XtInitialize[1]	XtSetSubvalues[1]
XtCallbackNonexclusive[1]	XtInitializeWidgetClass[1]	XtSetTypeConverter[1]
XtCallbackPopdown[1]	XtInsertEventHandler[1]	XtSetValues[1]
XtCallbackReleaseCacheRef[1]	XtInsertEventTypeHandler[1]	XtSetWMColormapWindows[1]
XtCallbackReleaseCacheRefList[1]	XtInsertRawEventHandler[1]	XtSetWarningHandler[1]
XtCalloc[1]	XtInstallAccelerators[1]	XtSetWarningMsgHandler[1]
XtCancelSelectionRequest[1]	XtInstallAllAccelerators[1]	XtStringConversionWarning[1]
XtChangeManagedSet[1]	XtIsApplicationShell[1]	XtSuperclass[1]
XtClass[1]	XtIsComposite[1]	XtToolkitInitialize[1]
XtCloseDisplay[1]	XtIsConstraint[1]	XtToolkitThreadInitialize[1]
XtConfigureWidget[1]	XtIsManaged[1]	XtTranslateCoords[1]
XtConvert[1]	XtIsObject[1]	XtTranslateKey[1]
XtConvertAndStore[1]	XtIsOverrideShell[1]	XtTranslateKeycode[1]
XtConvertCase[1]	XtIsRealized[1]	XtUngrabButton[1]
XtCreateApplicationContext[1]	XtIsRectObj[1]	XtUngrabKey[1]
XtCreateApplicationShell[1]	XtIsSensitive[1]	XtUngrabKeyboard[1]
XtCreateManagedWidget[1]	XtIsSessionShell[1]	XtUngrabPointer[1]
XtCreatePopupShell[1]	XtIsShell[1]	XtUninstallTranslations[1]
XtCreateSelectionRequest[1]	XtIsSubclass[1]	XtUnmanageChild[1]
XtCreateWidget[1]	XtIsTopLevelShell[1]	XtUnmanageChildren[1]
XtCreateWindow[1]	XtIsTransientShell[1]	XtUnmapWidget[1]
XtCvtColorToPixel[1]	XtIsVendorShell[1]	XtUnrealizeWidget[1]
XtCvtIntToBool[1]	XtIsWMShell[1]	XtUnregisterDrawable[1]
XtCvtIntToBoolean[1]	XtIsWidget[1]	XtVaAppCreateShell[1]
XtCvtIntToColor[1]	XtKeysymToKeycodeList[1]	XtVaAppInitialize[1]

XtCvtIntToFloat[1]	XtLastEventProcessed[1]	XtVaCreateArgsList[1]
XtCvtIntToFont[1]	XtLastTimestampProcessed[1]	XtVaCreateManagedWidget[1]
XtCvtIntToPixel[1]	XtMainLoop[1]	XtVaCreatePopupShell[1]
XtCvtIntToPixmap[1]	XtMakeGeometryRequest[1]	XtVaCreateWidget[1]
XtCvtIntToShort[1]	XtMakeResizeRequest[1]	XtVaGetApplicationResources[1]
XtCvtIntToUnsignedChar[1]	XtMalloc[1]	XtVaGetSubresources[1]
XtCvtStringToAcceleratorTable[1]	XtManageChild[1]	XtVaGetSubvalues[1]
XtCvtStringToAtom[1]	XtManageChildren[1]	XtVaGetValues[1]
XtCvtStringToBool[1]	XtMapWidget[1]	XtVaOpenApplication[1]
XtCvtStringToBoolean[1]	XtMenuPopupAction[1]	XtVaSetSubvalues[1]
XtCvtStringToCommandArgArray[1]	XtMergeArgLists[1]	XtVaSetValues[1]
XtCvtStringToCursor[1]	XtMoveWidget[1]	XtWarning[1]
XtCvtStringToDimension[1]	XtName[1]	XtWarningMsg[1]
XtCvtStringToDirectoryString[1]	XtNameToWidget[1]	XtWidgetToApplicationContext[1]
XtCvtStringToDisplay[1]	XtNewString[1]	XtWindow[1]
XtCvtStringToFile[1]	XtNextEvent[1]	XtWindowOfObject[1]
XtCvtStringToFloat[1]	XtNoticeSignal[1]	XtWindowToWidget[1]
XtCvtStringToFont[1]	XtOpenApplication[1]	_XtCheckSubclassFlag[1]
XtCvtStringToFontSet[1]	XtOpenDisplay[1]	_XtCopyFromArg[1]
XtCvtStringToFontStruct[1]	XtOverrideTranslations[1]	_XtInherit[1]
XtCvtStringToGravity[1]	XtOwnSelection[1]	_XtIsSubclassOf[1]

Table A-4. libXt Data Interfaces

XtCXtToolkitError	objectClass	topLevelShellClassRec
XtShellStrings	objectClassRec	topLevelShellWidgetClass
XtStrings	overrideShellClassRec	transientShellClassRec
_XtInheritTranslations	overrideShellWidgetClass	transientShellWidgetClass
applicationShellWidgetClass	rectObjClass	widgetClass
compositeClassRec	rectObjClassRec	widgetClassRec
compositeWidgetClass	sessionShellClassRec	wmShellClassRec
constraintClassRec	sessionShellWidgetClass	wmShellWidgetClass

<code>constraintWidgetClass</code>	<code>shellClassRee</code>	
<code>coreWidgetClass</code>	<code>shellWidgetClass</code>	

A.3. libmldl

The behaviour of the interfaces in this library is specified by the following Standards.

ISO/IEC 9899: 1999, *Programming Language C*—this specification
 CAE Specification, January 1997, *System Interfaces and Headers (XSH)*, Issue 5 (ISBN: 1 85912 181 0, C606) ISO
 POSIX (2003)
 ISO/IEC 9945:2003 *Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3*

Table A-4. libdl Function Interfaces

<code>dladdr(GLIBC_2.0)[1]</code>	<code>dlderror(GLIBC_2.0)[1]</code>	<code>dlsym(GLIBC_2.0)[1]</code>
<code>dlclose(GLIBC_2.0)[1]</code>	<code>dlopen(GLIBC_2.0)[1]</code>	

A.4. libm

The behaviour of the interfaces in this library is specified by the following Standards.

ISO C (1999)
 SUSv2
 ISO POSIX (2003)

Table A-5. libm Function Interfaces

<code>acos(GLIBC_2.0)</code> [1]	<code>csinh(GLIBC_2.0)</code> [1]	<code>log(GLIBC_2.0)</code> [1]
<code>acosf(GLIBC_2.0)</code> [1]	<code>csinl(GLIBC_2.0)</code> [1]	<code>log10(GLIBC_2.0)</code> [1]
<code>acosh(GLIBC_2.0)</code> [1]	<code>csqrt(GLIBC_2.0)</code> [1]	<code>log10f[1]</code>
<code>acoshf(GLIBC_2.0)</code> [1]	<code>csqrtf(GLIBC_2.0)</code> [1]	<code>log10l[1]</code>
<code>acoshl(GLIBC_2.0)</code> [1]	<code>csqrtl(GLIBC_2.0)</code> [1]	<code>log1p(GLIBC_2.0)</code> [1]
<code>acosl(GLIBC_2.0)</code> [1]	<code>ctan(GLIBC_2.0)</code> [1]	<code>logb(GLIBC_2.0)</code> [1]
<code>asin(GLIBC_2.0)</code> [1]	<code>ctanf(GLIBC_2.0)</code> [1]	<code>logflogf[1]</code>
<code>asinf(GLIBC_2.0)</code> [1]	<code>ctanh(GLIBC_2.0)</code> [1]	<code>logllogl[1]</code>

<code>asinh(GLIBC_2.0)</code> <code>asinh(GLIBC_2.0)[1]</code>	<code>ctanhf(GLIBC_2.0)</code> <code>ctanhf(GLIBC_2.0)[1]</code>	<code>lrint(GLIBC_2.0)</code> <code>lrint(GLIBC_2.0)[1]</code>
<code>asinhf(GLIBC_2.0)</code> <code>asinhf(GLIBC_2.0)[1]</code>	<code>ctanhl(GLIBC_2.0)</code> <code>ctanhl(GLIBC_2.0)[1]</code>	<code>lrintf(GLIBC_2.0)</code> <code>lrintf(GLIBC_2.0)[1]</code>
<code>asinhl(GLIBC_2.0)</code> <code>asinhl(GLIBC_2.0)[1]</code>	<code>ctanl(GLIBC_2.0)</code> <code>ctanl(GLIBC_2.0)[1]</code>	<code>lrintl(GLIBC_2.0)</code> <code>lrintl(GLIBC_2.0)[1]</code>
<code>asinl(GLIBC_2.0)</code> <code>asinl(GLIBC_2.0)[1]</code>	<code>dremf(GLIBC_2.0)</code> <code>dremf(GLIBC_2.0)[1]</code>	<code>lround(GLIBC_2.0)</code> <code>lround(GLIBC_2.0)[1]</code>
<code>atan(GLIBC_2.0)</code> <code>atan(GLIBC_2.0)[1]</code>	<code>dremf(GLIBC_2.0)</code> <code>dremf(GLIBC_2.0)[1]</code>	<code>lroundf(GLIBC_2.0)</code> <code>lroundf(GLIBC_2.0)[1]</code>
<code>atan2(GLIBC_2.0)</code> <code>atan2(GLIBC_2.0)[1]</code>	<code>erf(GLIBC_2.0)</code> <code>erf(GLIBC_2.0)[1]</code>	<code>lroundl(GLIBC_2.0)</code> <code>lroundl(GLIBC_2.0)[1]</code>
<code>atan2f(GLIBC_2.0)</code> <code>atan2f(GLIBC_2.0)[1]</code>	<code>erfc(GLIBC_2.0)</code> <code>erfc(GLIBC_2.0)[1]</code>	<code>matherr(GLIBC_2.0)</code> <code>matherr(GLIBC_2.0)[1]</code>
<code>atan2l(GLIBC_2.0)</code> <code>atan2l(GLIBC_2.0)[1]</code>	<code>erfcf(GLIBC_2.0)</code> <code>erfcf(GLIBC_2.0)[1]</code>	<code>modf(GLIBC_2.0)</code> <code>modf(GLIBC_2.0)[1]</code>
<code>atanf(GLIBC_2.0)</code> <code>atanf(GLIBC_2.0)[1]</code>	<code>erfcf(GLIBC_2.0)</code> <code>erfcf(GLIBC_2.0)[1]</code>	<code>modff(GLIBC_2.0)</code> <code>modff(GLIBC_2.0)[1]</code>
<code>atanh(GLIBC_2.0)</code> <code>atanh(GLIBC_2.0)[1]</code>	<code>erff(GLIBC_2.0)</code> <code>erff(GLIBC_2.0)[1]</code>	<code>modfl(GLIBC_2.0)</code> <code>modfl(GLIBC_2.0)[1]</code>
<code>atanhf(GLIBC_2.0)</code> <code>atanhf(GLIBC_2.0)[1]</code>	<code>erfl(GLIBC_2.0)</code> <code>erfl(GLIBC_2.0)[1]</code>	<code>nan(GLIBC_2.0)</code> <code>nan(GLIBC_2.0)[1]</code>
<code>atanhl(GLIBC_2.0)</code> <code>atanhl(GLIBC_2.0)[1]</code>	<code>exp(GLIBC_2.0)</code> <code>exp(GLIBC_2.0)[1]</code>	<code>nanf(GLIBC_2.0)</code> <code>nanf(GLIBC_2.0)[1]</code>
<code>atanl(GLIBC_2.0)</code> <code>atanl(GLIBC_2.0)[1]</code>	<code>expf</code> <code>expf[1]</code>	<code>nanl(GLIBC_2.0)</code> <code>nanl(GLIBC_2.0)[1]</code>
<code>cabs(GLIBC_2.1)</code> <code>cabs(GLIBC_2.1)[1]</code>	<code>exp</code> <code>exp[1]</code>	<code>nearbyint(GLIBC_2.1)</code> <code>nearbyint(GLIBC_2.1)[1]</code>
<code>cabsf(GLIBC_2.1)</code> <code>cabsf(GLIBC_2.1)[1]</code>	<code>expm1(GLIBC_2.1)</code> <code>expm1(GLIBC_2.1)[1]</code>	<code>nearbyintf(GLIBC_2.1)</code> <code>nearbyintf(GLIBC_2.1)[1]</code>
<code>cabsl(GLIBC_2.1)</code> <code>cabsl(GLIBC_2.1)[1]</code>	<code>fabs(GLIBC_2.1)</code> <code>fabs(GLIBC_2.1)[1]</code>	<code>nearbyintl(GLIBC_2.1)</code> <code>nearbyintl(GLIBC_2.1)[1]</code>
<code>cacos(GLIBC_2.1)</code> <code>cacos(GLIBC_2.1)[1]</code>	<code>fabsf(GLIBC_2.1)</code> <code>fabsf(GLIBC_2.1)[1]</code>	<code>nextafter(GLIBC_2.1)</code> <code>nextafter(GLIBC_2.1)[1]</code>
<code>cacosf(GLIBC_2.1)</code> <code>cacosf(GLIBC_2.1)[1]</code>	<code>fabsl(GLIBC_2.1)</code> <code>fabsl(GLIBC_2.1)[1]</code>	<code>nextafterf(GLIBC_2.1)</code> <code>nextafterf(GLIBC_2.1)[1]</code>
<code>cacosh(GLIBC_2.1)</code> <code>cacosh(GLIBC_2.1)[1]</code>	<code>fdim(GLIBC_2.1)</code> <code>fdim(GLIBC_2.1)[1]</code>	<code>nextafterl(GLIBC_2.1)</code> <code>nextafterl(GLIBC_2.1)[1]</code>

<code>_2.1)[1]</code>	<code>)[1]</code>	<code>LIBC_2.1)[1]</code>
<code>cacoshf(GLIBC_2.1)cacoshf(GLIBC_2.1)[1]</code>	<code>fdimf(GLIBC_2.1)fdimf(GLIBC_2.1)[1]</code>	<code>nexttoward(GLIBC_2.1)nexttoward(GLIBC_2.1)[1]</code>
<code>cacoshl(GLIBC_2.1)cacoshl(GLIBC_2.1)[1]</code>	<code>fdiml(GLIBC_2.1)fdiml(GLIBC_2.1)[1]</code>	<code>nexttowardf(GLIBC_2.1)nexttowardf(GLIBC_2.1)[1]</code>
<code>cacosl(GLIBC_2.1)cacosl(GLIBC_2.1)[1]</code>	<code>feclearexcept(GLIBC_2.1)[1]</code>	<code>nexttowardl(GLIBC_2.1)nexttowardl(GLIBC_2.1)[1]</code>
<code>carg(GLIBC_2.1)carg(GLIBC_2.1)[1]</code>	<code>fegetenv(GLIBC_2.1)fegetenv(GLIBC_2.1)[1]</code>	<code>pow(GLIBC_2.1)pow(GLIBC_2.1)[1]</code>
<code>cargf(GLIBC_2.1)cargf(GLIBC_2.1)[1]</code>	<code>fegetexceptflag(GLIBC_2.1)[1]</code>	<code>pow10(GLIBC_2.1)pow10(GLIBC_2.1)[1]</code>
<code>cargl(GLIBC_2.1)cargl(GLIBC_2.1)[1]</code>	<code>fegetround(GLIBC_2.1)fegetround(GLIBC_2.1)[1]</code>	<code>pow10f(GLIBC_2.1)pow10f(GLIBC_2.1)[1]</code>
<code>casin(GLIBC_2.1)casin(GLIBC_2.1)[1]</code>	<code>feholdexcept(GLIBC_2.1)feholdexcept(GLIBC_2.1)[1]</code>	<code>pow10l(GLIBC_2.1)pow10l(GLIBC_2.1)[1]</code>
<code>casinf(GLIBC_2.1)casinf(GLIBC_2.1)[1]</code>	<code>feraiseexcept(GLIBC_2.1)[1]</code>	<code>powf(GLIBC_2.1)powf(GLIBC_2.1)[1]</code>
<code>casinh(GLIBC_2.1)casinh(GLIBC_2.1)[1]</code>	<code>fesetenv(GLIBC_2.1)fesetenv(GLIBC_2.1)[1]</code>	<code>powl(GLIBC_2.1)powl(GLIBC_2.1)[1]</code>
<code>casinhf(GLIBC_2.1)casinhf(GLIBC_2.1)[1]</code>	<code>fesetexceptflag(GLIBC_2.1)[1]</code>	<code>remainder(GLIBC_2.1)remainder(GLIBC_2.1)[1]</code>
<code>casinhl(GLIBC_2.1)casinhl(GLIBC_2.1)[1]</code>	<code>fesetround(GLIBC_2.1)fesetround(GLIBC_2.1)[1]</code>	<code>remainderf(GLIBC_2.1)remainderf(GLIBC_2.1)[1]</code>
<code>casinl(GLIBC_2.1)casinl(GLIBC_2.1)[1]</code>	<code>fetestexcept(GLIBC_2.1)fetestexcept(GLIBC_2.1)[1]</code>	<code>remainderl(GLIBC_2.1)remainderl(GLIBC_2.1)[1]</code>
<code>catan(GLIBC_2.1)catan(GLIBC_2.1)[1]</code>	<code>feupdateenv(GLIBC_2.1)feupdateenv(GLIBC_2.1)[1]</code>	<code>remquo(GLIBC_2.1)remquo(GLIBC_2.1)[1]</code>
<code>catanf(GLIBC_2.1)catanf(GLIBC_2.1)[1]</code>	<code>finite(GLIBC_2.1)finite(GLIBC_2.1)[1]</code>	<code>remquof(GLIBC_2.1)remquof(GLIBC_2.1)[1]</code>
<code>catanh(GLIBC_2.1)catanh(GLIBC_2.1)[1]</code>	<code>finitef(GLIBC_2.1)finitef(GLIBC_2.1)[1]</code>	<code>remquol(GLIBC_2.1)remquol(GLIBC_2.1)[1]</code>
<code>catanhf(GLIBC_2.1)catanhf(GLIBC_2.1)[1]</code>	<code>finitel(GLIBC_2.1)finitel(GLIBC_2.1)[1]</code>	<code>rint(GLIBC_2.1)rint(GLIBC_2.1)[1]</code>
<code>catanhl(GLIBC_2.1)catanhl(GLIBC_2.1)[1]</code>	<code>floor(GLIBC_2.1)floor(GLIBC_2.1)[1]</code>	<code>rintf(GLIBC_2.1)rintf(GLIBC_2.1)[1]</code>
<code>catanl(GLIBC_2.1)catanl(GLIBC_2.1)[1]</code>	<code>floorf(GLIBC_2.1)floorf(GLIBC_2.1)[1]</code>	<code>rintl(GLIBC_2.1)rintl(GLIBC_2.1)[1]</code>

<code>cbrt(GLIBC_2.0)cbrt(GLIBC_2.0)[1]</code>	<code>floorl(GLIBC_2.0)floorl(GLIBC_2.0)[1]</code>	<code>round(GLIBC_2.0)round(GLIBC_2.0)[1]</code>
<code>cbrtf(GLIBC_2.0)cbrtf(GLIBC_2.0)[1]</code>	<code>fma(GLIBC_2.0)fma(GLIBC_2.0)[1]</code>	<code>roundf(GLIBC_2.0)roundf(GLIBC_2.0)[1]</code>
<code>cbrtl(GLIBC_2.0)cbrtl(GLIBC_2.0)[1]</code>	<code>fmaf(GLIBC_2.0)fmaf(GLIBC_2.0)[1]</code>	<code>roundl(GLIBC_2.0)roundl(GLIBC_2.0)[1]</code>
<code>ccos(GLIBC_2.1)ccos(GLIBC_2.1)[1]</code>	<code>fmal(GLIBC_2.1)fmal(GLIBC_2.1)[1]</code>	<code>scalb(GLIBC_2.1)scalb(GLIBC_2.1)[1]</code>
<code>ccosf(GLIBC_2.1)ccosf(GLIBC_2.1)[1]</code>	<code>fmax(GLIBC_2.1)fmax(GLIBC_2.1)[1]</code>	<code>scalbf(GLIBC_2.1)scalbf(GLIBC_2.1)[1]</code>
<code>ccosh(GLIBC_2.1)ccosh(GLIBC_2.1)[1]</code>	<code>fmaxf(GLIBC_2.1)fmaxf(GLIBC_2.1)[1]</code>	<code>scalbl(GLIBC_2.1)scalbl(GLIBC_2.1)[1]</code>
<code>ccoshf(GLIBC_2.1)ccoshf(GLIBC_2.1)[1]</code>	<code>fmaxl(GLIBC_2.1)fmaxl(GLIBC_2.1)[1]</code>	<code>scalbln(GLIBC_2.1)scalbln(GLIBC_2.1)[1]</code>
<code>ccoshl(GLIBC_2.1)ccoshl(GLIBC_2.1)[1]</code>	<code>fmin(GLIBC_2.1)fmin(GLIBC_2.1)[1]</code>	<code>scalblnf(GLIBC_2.1)scalblnf(GLIBC_2.1)[1]</code>
<code>ccosl(GLIBC_2.1)ccosl(GLIBC_2.1)[1]</code>	<code>fminf(GLIBC_2.1)fminf(GLIBC_2.1)[1]</code>	<code>scalblnl(GLIBC_2.1)scalblnl(GLIBC_2.1)[1]</code>
<code>ceil(GLIBC_2.0)ceil(GLIBC_2.0)[1]</code>	<code>fminl(GLIBC_2.0)fminl(GLIBC_2.0)[1]</code>	<code>scalbn(GLIBC_2.0)scalbn(GLIBC_2.0)[1]</code>
<code>ceilf(GLIBC_2.0)ceilf(GLIBC_2.0)[1]</code>	<code>fmod(GLIBC_2.0)fmod(GLIBC_2.0)[1]</code>	<code>scalbnf(GLIBC_2.0)scalbnf(GLIBC_2.0)[1]</code>
<code>ceill(GLIBC_2.0)ceill(GLIBC_2.0)[1]</code>	<code>fmodf(GLIBC_2.0)fmodf(GLIBC_2.0)[1]</code>	<code>scalbnl(GLIBC_2.0)scalbnl(GLIBC_2.0)[1]</code>
<code>cexp(GLIBC_2.1)cexp(GLIBC_2.1)[1]</code>	<code>fmodl(GLIBC_2.1)fmodl(GLIBC_2.1)[1]</code>	<code>significand(GLIBC_2.1)significand(GLIBC_2.1)[1]</code>
<code>cexpf(GLIBC_2.1)cexpf(GLIBC_2.1)[1]</code>	<code>frexp(GLIBC_2.1)frexp(GLIBC_2.1)[1]</code>	<code>significandf(GLIBC_2.1)significandf(GLIBC_2.1)[1]</code>
<code>cexpl(GLIBC_2.1)cexpl(GLIBC_2.1)[1]</code>	<code>frexpf(GLIBC_2.1)frexpf(GLIBC_2.1)[1]</code>	<code>significandl(GLIBC_2.1)significandl(GLIBC_2.1)[1]</code>
<code>cimag(GLIBC_2.1)cimag(GLIBC_2.1)[1]</code>	<code>frexpl(GLIBC_2.1)frexpl(GLIBC_2.1)[1]</code>	<code>sin(GLIBC_2.1)sin(GLIBC_2.1)[1]</code>
<code>cimagf(GLIBC_2.1)cimagf(GLIBC_2.1)[1]</code>	<code>gamma(GLIBC_2.1)gamma(GLIBC_2.1)[1]</code>	<code>sincos(GLIBC_2.1)sincos(GLIBC_2.1)[1]</code>
<code>cimagl(GLIBC_2.1)cimagl(GLIBC_2.1)[1]</code>	<code>gammaf(GLIBC_2.1)gammaf(GLIBC_2.1)[1]</code>	<code>sincosf(GLIBC_2.1)sincosf(GLIBC_2.1)[1]</code>
<code>clog(GLIBC_2.1)clog(GLIBC_2.1)</code>	<code>gammal(GLIBC_2.1)gammal(GLIBC_2.1)</code>	<code>sincosl(GLIBC_2.1)sincosl(GLIBC_2.1)</code>

[1]	BC_2.1)[1]	_2.1)[1]
eelog10(GLIBC_2.1)clog10(GLIBC_2.1)[1]	hypot(GLIBC_2.1)hypot(GLIBC_2.1)[1]	sinf(GLIBC_2.1)sinf(GLIBC_2.1)[1]
eelog10f(GLIBC_2.1)clog10f(GLIBC_2.1)[1]	hypotf(GLIBC_2.1)hypotf(GLIBC_2.1)[1]	sinh(GLIBC_2.1)sinh(GLIBC_2.1)[1]
eelog10l(GLIBC_2.1)clog10l(GLIBC_2.1)[1]	hypotl(GLIBC_2.1)hypotl(GLIBC_2.1)[1]	sinhf(GLIBC_2.1)sinhf(GLIBC_2.1)[1]
eelogf(GLIBC_2.1)clogf(GLIBC_2.1)[1]	ilogb(GLIBC_2.1)ilogb(GLIBC_2.1)[1]	sinhl(GLIBC_2.1)sinhl(GLIBC_2.1)[1]
eelogl(GLIBC_2.1)clogl(GLIBC_2.1)[1]	ilogbf(GLIBC_2.1)ilogbf(GLIBC_2.1)[1]	sinl(GLIBC_2.1)sinl(GLIBC_2.1)[1]
conj(GLIBC_2.1)conj(GLIBC_2.1)[1]	ilogbl(GLIBC_2.1)ilogbl(GLIBC_2.1)[1]	sqrt(GLIBC_2.1)sqrt(GLIBC_2.1)[1]
conjf(GLIBC_2.1)conjf(GLIBC_2.1)[1]	j0(GLIBC_2.1)j0(GLIBC_2.1)[1]	sqrtf(GLIBC_2.1)sqrtf(GLIBC_2.1)[1]
conjl(GLIBC_2.1)conjl(GLIBC_2.1)[1]	j0f(GLIBC_2.1)j0f(GLIBC_2.1)[1]	sqrtl(GLIBC_2.1)sqrtl(GLIBC_2.1)[1]
copysign(GLIBC_2.0)copysign(GLIBC_2.0)[1]	j0l(GLIBC_2.0)j0l(GLIBC_2.0)[1]	tan(GLIBC_2.0)tan(GLIBC_2.0)[1]
copysignf(GLIBC_2.0)copysignf(GLIBC_2.0)[1]	j1(GLIBC_2.0)j1(GLIBC_2.0)[1]	tanf(GLIBC_2.0)tanf(GLIBC_2.0)[1]
copysignl(GLIBC_2.0)copysignl(GLIBC_2.0)[1]	j1f(GLIBC_2.0)j1f(GLIBC_2.0)[1]	tanh(GLIBC_2.0)tanh(GLIBC_2.0)[1]
cos(GLIBC_2.0)cos(GLIBC_2.0)[1]	j1l(GLIBC_2.0)j1l(GLIBC_2.0)[1]	tanhf(GLIBC_2.0)tanhf(GLIBC_2.0)[1]
cosf(GLIBC_2.0)cosf(GLIBC_2.0)[1]	jn(GLIBC_2.0)jn(GLIBC_2.0)[1]	tanhhl(GLIBC_2.0)tanhhl(GLIBC_2.0)[1]
cosh(GLIBC_2.0)cosh(GLIBC_2.0)[1]	jnf(GLIBC_2.0)jnf(GLIBC_2.0)[1]	tanl(GLIBC_2.0)tanl(GLIBC_2.0)[1]
coshf(GLIBC_2.0)coshf(GLIBC_2.0)[1]	jnl(GLIBC_2.0)jnl(GLIBC_2.0)[1]	tgamma(GLIBC_2.0)tgamma(GLIBC_2.0)[1]
coshl(GLIBC_2.0)coshl(GLIBC_2.0)[1]	ldexp(GLIBC_2.0)ldexp(GLIBC_2.0)[1]	tgammaf(GLIBC_2.0)tgammaf(GLIBC_2.0)[1]
cosl(GLIBC_2.0)cosl(GLIBC_2.0)[1]	ldexpf(GLIBC_2.0)ldexpf(GLIBC_2.0)[1]	tgammal(GLIBC_2.0)tgammal(GLIBC_2.0)[1]
cpow(GLIBC_2.1)cpow(GLIBC_2.1)[1]	ldexpl(GLIBC_2.1)ldexpl(GLIBC_2.1)[1]	trunc(GLIBC_2.1)trunc(GLIBC_2.1)[1]

<code>epowf(GLIBC_2.1)cpowf(GLIBC_2.1)[1]</code>	<code>lgamma(GLIBC_2.1)lgamma(GLIBC_2.1)[1]</code>	<code>truncf(GLIBC_2.1)truncf(GLIBC_2.1)[1]</code>
<code>epowl(GLIBC_2.1)cpowl(GLIBC_2.1)[1]</code>	<code>lgamma_r(GLIBC_2.1)lgamma_r(GLIBC_2.1)[1]</code>	<code>truncr(GLIBC_2.1)truncr(GLIBC_2.1)[1]</code>
<code>eprojl(GLIBC_2.1)cproj(GLIBC_2.1)[1]</code>	<code>lgammaf(GLIBC_2.1)lgammaf(GLIBC_2.1)[1]</code>	<code>y0(GLIBC_2.1)y0(GLIBC_2.1)[1]</code>
<code>eprojf(GLIBC_2.1)cprojf(GLIBC_2.1)[1]</code>	<code>lgammaf_r(GLIBC_2.1)lgammaf_r(GLIBC_2.1)[1]</code>	<code>y0f(GLIBC_2.1)y0f(GLIBC_2.1)[1]</code>
<code>eprojl(GLIBC_2.1)cprojl(GLIBC_2.1)[1]</code>	<code>lgammal(GLIBC_2.1)lgammal(GLIBC_2.1)[1]</code>	<code>y0l(GLIBC_2.1)y0l(GLIBC_2.1)[1]</code>
<code>erealf(GLIBC_2.1)crealf(GLIBC_2.1)[1]</code>	<code>lgammal_r(GLIBC_2.1)lgammal_r(GLIBC_2.1)[1]</code>	<code>y1(GLIBC_2.1)y1(GLIBC_2.1)[1]</code>
<code>erealf(GLIBC_2.1)crealf(GLIBC_2.1)[1]</code>	<code>llrint(GLIBC_2.1)llrint(GLIBC_2.1)[1]</code>	<code>y1f(GLIBC_2.1)y1f(GLIBC_2.1)[1]</code>
<code>ereall(GLIBC_2.1)creall(GLIBC_2.1)[1]</code>	<code>llrintf(GLIBC_2.1)llrintf(GLIBC_2.1)[1]</code>	<code>y1l(GLIBC_2.1)y1l(GLIBC_2.1)[1]</code>
<code>esin(GLIBC_2.1)csin(GLIBC_2.1)[1]</code>	<code>llrintl(GLIBC_2.1)llrintl(GLIBC_2.1)[1]</code>	<code>yn(GLIBC_2.1)yn(GLIBC_2.1)[1]</code>
<code>esinf(GLIBC_2.1)csinf(GLIBC_2.1)[1]</code>	<code>llround(GLIBC_2.1)llround(GLIBC_2.1)[1]</code>	<code>ynf(GLIBC_2.1)ynf(GLIBC_2.1)[1]</code>
<code>esinh(GLIBC_2.1)csinh(GLIBC_2.1)[1]</code>	<code>llroundf(GLIBC_2.1)llroundf(GLIBC_2.1)[1]</code>	<code>ynl(GLIBC_2.1)ynl(GLIBC_2.1)[1]</code>
<code>esinhf(GLIBC_2.1)csinhf(GLIBC_2.1)[1]</code>	<code>llroundl(GLIBC_2.1)llroundl(GLIBC_2.1)[1]</code>	

Table A-6. libm Data Interfaces

<code>signgam</code> <code>signgamID_STD_46_SUS_V3</code>		
--	--	--

A.4. libGL

The behaviour of the interfaces in this library is specified by the following Standards.

Table A-7. libGL Function Interfaces

<code>glAccum[1]</code>	<code>glGetString[1]</code>	<code>glRasterPos4iv[1]</code>
-------------------------	-----------------------------	--------------------------------

glActiveTextureARB[1]	glGetTexEnvfv[1]	glRasterPos4s[1]
glAlphaFunc[1]	glGetTexEnviv[1]	glRasterPos4sv[1]
glAreTexturesResident[1]	glGetTexGendv[1]	glReadBuffer[1]
glArrayElement[1]	glGetTexGenfv[1]	glReadPixels[1]
glBegin[1]	glGetTexGeniv[1]	glReectd[1]
glBindTexture[1]	glGetTexImage[1]	glReectdv[1]
glBitmap[1]	glGetTexLevelParameterfv[1]	glRectf[1]
glBlendColor[1]	glGetTexLevelParameteriv[1]	glReectfv[1]
glBlendEquation[1]	glGetTexParameterfv[1]	glRecti[1]
glBlendFunc[1]	glGetTexParameteriv[1]	glRectiv[1]
glCallList[1]	glHint[1]	glReects[1]
glCallLists[1]	glHistogram[1]	glReectsv[1]
glClear[1]	glIndexMask[1]	glRenderMode[1]
glClearAccum[1]	glIndexPointer[1]	glResetHistogram[1]
glClearColor[1]	glIndexd[1]	glResetMinmax[1]
glClearDepth[1]	glIndexdv[1]	glRotated[1]
glClearIndex[1]	glIndexf[1]	glRotatef[1]
glClearStencil[1]	glIndexfv[1]	glScaled[1]
glClientActiveTextureARB[1]	glIndexi[1]	glScalef[1]
glClipPlane[1]	glIndexiv[1]	glScissor[1]
glColor3b[1]	glIndexs[1]	glSelectBuffer[1]
glColor3bv[1]	glIndexsv[1]	glSeparableFilter2D[1]
glColor3d[1]	glIndexub[1]	glShadeModel[1]
glColor3dv[1]	glIndexubv[1]	glStencilFunc[1]
glColor3f[1]	glInitNames[1]	glStencilMask[1]
glColor3fv[1]	glInterleavedArrays[1]	glStencilOp[1]
glColor3i[1]	glIsEnabled[1]	glTexCoord1d[1]
glColor3iv[1]	glIsList[1]	glTexCoord1dv[1]
glColor3s[1]	glIsTexture[1]	glTexCoord1f[1]
glColor3sv[1]	glLightModelf[1]	glTexCoord1fv[1]
glColor3ub[1]	glLightModelfv[1]	glTexCoord1i[1]

glColor3ubv[1]	glLightModeli[1]	glTexCoord1iv[1]
glColor3ui[1]	glLightModeliv[1]	glTexCoord1s[1]
glColor3uiv[1]	glLightf[1]	glTexCoord1sv[1]
glColor3us[1]	glLightfv[1]	glTexCoord2d[1]
glColor3usv[1]	glLighti[1]	glTexCoord2dv[1]
glColor4b[1]	glLightiv[1]	glTexCoord2f[1]
glColor4bv[1]	glLineStipple[1]	glTexCoord2fv[1]
glColor4d[1]	glLineWidth[1]	glTexCoord2i[1]
glColor4dv[1]	glListBase[1]	glTexCoord2iv[1]
glColor4f[1]	glLoadIdentity[1]	glTexCoord2s[1]
glColor4fv[1]	glLoadMatrixd[1]	glTexCoord2sv[1]
glColor4i[1]	glLoadMatrixf[1]	glTexCoord3d[1]
glColor4iv[1]	glLoadName[1]	glTexCoord3dv[1]
glColor4s[1]	glLogicOp[1]	glTexCoord3f[1]
glColor4sv[1]	glMap1d[1]	glTexCoord3fv[1]
glColor4ub[1]	glMap1f[1]	glTexCoord3i[1]
glColor4ubv[1]	glMap2d[1]	glTexCoord3iv[1]
glColor4ui[1]	glMap2f[1]	glTexCoord3s[1]
glColor4uiv[1]	glMapGrid1d[1]	glTexCoord3sv[1]
glColor4us[1]	glMapGrid1f[1]	glTexCoord4d[1]
glColor4usv[1]	glMapGrid2d[1]	glTexCoord4dv[1]
glColorMask[1]	glMapGrid2f[1]	glTexCoord4f[1]
glColorMaterial[1]	glMaterialf[1]	glTexCoord4fv[1]
glColorPointer[1]	glMaterialfv[1]	glTexCoord4i[1]
glColorSubTable[1]	glMateriali[1]	glTexCoord4iv[1]
glColorTable[1]	glMaterialiv[1]	glTexCoord4s[1]
glColorTableParameterfv[1]	glMatrixMode[1]	glTexCoord4sv[1]
glColorTableParameteriv[1]	glMinmax[1]	glTexCoordPointer[1]
glConvolutionFilter1D[1]	glMultMatrixd[1]	glTexEnvf[1]
glConvolutionFilter2D[1]	glMultMatrixf[1]	glTexEnvfv[1]
glConvolutionParameterf[1]	glMultiTexCoord1dARB[1]	glTexEnvi[1]

glConvolutionParameterfv[1]	glMultiTexCoord1dvARB[1]	glTexEnviv[1]
glConvolutionParameteri[1]	glMultiTexCoord1fARB[1]	glTexGend[1]
glConvolutionParameteriv[1]	glMultiTexCoord1fvARB[1]	glTexGendv[1]
glCopyColorSubTable[1]	glMultiTexCoord1iARB[1]	glTexGenf[1]
glCopyColorTable[1]	glMultiTexCoord1ivARB[1]	glTexGenfv[1]
glCopyConvolutionFilter1D[1]	glMultiTexCoord1sARB[1]	glTexGeni[1]
glCopyConvolutionFilter2D[1]	glMultiTexCoord1svARB[1]	glTexGeniv[1]
glCopyPixels[1]	glMultiTexCoord2dARB[1]	glTexImage1D[1]
glCopyTexImage1D[1]	glMultiTexCoord2dvARB[1]	glTexImage2D[1]
glCopyTexImage2D[1]	glMultiTexCoord2fARB[1]	glTexImage3D[1]
glCopyTexSubImage1D[1]	glMultiTexCoord2fvARB[1]	glTexParameterf[1]
glCopyTexSubImage2D[1]	glMultiTexCoord2iARB[1]	glTexParameterfv[1]
glCopyTexSubImage3D[1]	glMultiTexCoord2ivARB[1]	glTexParameteri[1]
glCullFace[1]	glMultiTexCoord2sARB[1]	glTexParameteriv[1]
glDeleteLists[1]	glMultiTexCoord2svARB[1]	glTexSubImage1D[1]
glDeleteTextures[1]	glMultiTexCoord3dARB[1]	glTexSubImage2D[1]
glDepthFunc[1]	glMultiTexCoord3dvARB[1]	glTexSubImage3D[1]
glDepthMask[1]	glMultiTexCoord3fARB[1]	glTranslated[1]
glDepthRange[1]	glMultiTexCoord3fvARB[1]	glTranslatef[1]
glDisable[1]	glMultiTexCoord3iARB[1]	glVertex2d[1]
glDisableClientState[1]	glMultiTexCoord3ivARB[1]	glVertex2dv[1]
glDrawArrays[1]	glMultiTexCoord3sARB[1]	glVertex2f[1]
glDrawBuffer[1]	glMultiTexCoord3svARB[1]	glVertex2fv[1]
glDrawElements[1]	glMultiTexCoord4dARB[1]	glVertex2i[1]
glDrawPixels[1]	glMultiTexCoord4dvARB[1]	glVertex2iv[1]
glDrawRangeElements[1]	glMultiTexCoord4fARB[1]	glVertex2s[1]
glEdgeFlag[1]	glMultiTexCoord4fvARB[1]	glVertex2sv[1]
glEdgeFlagPointer[1]	glMultiTexCoord4iARB[1]	glVertex3d[1]
glEdgeFlagv[1]	glMultiTexCoord4ivARB[1]	glVertex3dv[1]
glEnable[1]	glMultiTexCoord4sARB[1]	glVertex3f[1]
glEnableClientState[1]	glMultiTexCoord4svARB[1]	glVertex3fv[1]

glEnd[1]	glNewList[1]	glVertex3i[1]
glEndList[1]	glNormal3b[1]	glVertex3iv[1]
glEvalCoord1d[1]	glNormal3bv[1]	glVertex3s[1]
glEvalCoord1dv[1]	glNormal3d[1]	glVertex3sv[1]
glEvalCoord1f[1]	glNormal3dv[1]	glVertex4d[1]
glEvalCoord1fv[1]	glNormal3f[1]	glVertex4dv[1]
glEvalCoord2d[1]	glNormal3fv[1]	glVertex4f[1]
glEvalCoord2dv[1]	glNormal3i[1]	glVertex4fv[1]
glEvalCoord2f[1]	glNormal3iv[1]	glVertex4i[1]
glEvalCoord2fv[1]	glNormal3s[1]	glVertex4iv[1]
glEvalMesh1[1]	glNormal3sv[1]	glVertex4s[1]
glEvalMesh2[1]	glNormalPointer[1]	glVertex4sv[1]
glEvalPoint1[1]	glOrtho[1]	glVertexPointer[1]
glEvalPoint2[1]	glPassThrough[1]	glViewport[1]
glFeedbackBuffer[1]	glPixelMapfv[1]	glXChooseFBConfig[1]
glFinish[1]	glPixelMapuiv[1]	glXChooseVisual[1]
glFlush[1]	glPixelMapusv[1]	glXCopyContext[1]
glFogf[1]	glPixelStoref[1]	glXCreateContext[1]
glFogfv[1]	glPixelStorei[1]	glXCreateGLXPixmap[1]
glFogi[1]	glPixelTransferf[1]	glXCreateNewContext[1]
glFogiv[1]	glPixelTransferi[1]	glXCreatePbuffer[1]
glFrontFace[1]	glPixelZoom[1]	glXCreatePixmap[1]
glFrustum[1]	glPointSize[1]	glXCreateWindow[1]
glGenLists[1]	glPolygonMode[1]	glXDestroyContext[1]
glGenTextures[1]	glPolygonOffset[1]	glXDestroyGLXPixmap[1]
glGetBooleanv[1]	glPolygonStipple[1]	glXDestroyPbuffer[1]
glGetClipPlane[1]	glPopAttrib[1]	glXDestroyPixmap[1]
glGetColorTable[1]	glPopClientAttrib[1]	glXDestroyWindow[1]
glGetColorTableParameterfv[1]	glPopMatrix[1]	glXFreeContextEXT[1]
glGetColorTableParameteriv[1]	glPopName[1]	glXGetClientString[1]
glGetConvolutionFilter[1]	glPrioritizeTextures[1]	glXGetConfig[1]

glGetConvolutionParameterfv[1]	glPushAttrib[1]	glXGetContextIDEXT[1]
glGetConvolutionParameteriv[1]	glPushClientAttrib[1]	glXGetCurrentContext[1]
glGetDoublev[1]	glPushMatrix[1]	glXGetCurrentDisplay[1]
glGetError[1]	glPushName[1]	glXGetCurrentDrawable[1]
glGetFloatv[1]	glRasterPos2d[1]	glXGetCurrentReadDrawable[1]
glGetHistogram[1]	glRasterPos2dv[1]	glXGetFBConfigAttrib[1]
glGetHistogramParameterfv[1]	glRasterPos2f[1]	glXGetProcAddressARB[1]
glGetHistogramParameteriv[1]	glRasterPos2fv[1]	glXGetSelectedEvent[1]
glGetIntegerv[1]	glRasterPos2i[1]	glXGetVisualFromFBConfig[1]
glGetLightfv[1]	glRasterPos2iv[1]	glXImportContextEXT[1]
glGetLightiv[1]	glRasterPos2s[1]	glXIsDirect[1]
glGetMapdv[1]	glRasterPos2sv[1]	glXMakeContextCurrent[1]
glGetMapfv[1]	glRasterPos3d[1]	glXMakeCurrent[1]
glGetMapiv[1]	glRasterPos3dv[1]	glXQueryContext[1]
glGetMaterialfv[1]	glRasterPos3f[1]	glXQueryContextInfoEXT[1]
glGetMaterialiv[1]	glRasterPos3fv[1]	glXQueryDrawable[1]
glGetMinmax[1]	glRasterPos3i[1]	glXQueryExtension[1]
glGetMinmaxParameterfv[1]	glRasterPos3iv[1]	glXQueryExtensionsString[1]
glGetMinmaxParameteriv[1]	glRasterPos3s[1]	glXQueryServerString[1]
glGetPixelMapfv[1]	glRasterPos3sv[1]	glXQueryVersion[1]
glGetPixelMapuiv[1]	glRasterPos4d[1]	glXSelectEvent[1]
glGetPixelMapusv[1]	glRasterPos4dv[1]	glXSwapBuffers[1]
glGetPointerv[1]	glRasterPos4f[1]	glXUseXFont[1]
glGetPolygonStipple[1]	glRasterPos4fv[1]	glXWaitGL[1]
glGetSeparableFilter[1]	glRasterPos4i[1]	glXWaitX[1]

A.5. libXextlibncurses

The behaviour of the interfaces in this library is specified by the following Standards.

X/Open Curses

Table A-8. libXext7. libncurses Function Interfaces

addch[1]	mvdelch[1]	slk_refresh[1]
addchnstr[1]	mvderwin[1]	slk_restore[1]
addchstr[1]	mvgetch[1]	slk_set[1]
addnstr[1]	mvgetnstr[1]	slk_touch[1]
addstr[1]	mvgetstr[1]	standend[1]
DPMSCapable[1]attr_get[1]	XShmCreateImage[1]mvhline[1]	XSyncQueryExtension[1]standout[1]
DPMSDisable[1]attr_off[1]	XShmCreatePixmap[1]mvinch[1]	XSyncSetCounter[1]start_color[1]
DPMSEnable[1]attr_on[1]	XShmDetach[1]mvinchnstr[1]	XSyncSetPriority[1]subpad[1]
DPMSForceLevel[1]attr_set[1]	XShmGetEventBase[1]mvinchstr[1]	XSyncValueAdd[1]subwin[1]
DPMSGetTimeouts[1]attroff[1]	XShmGetImage[1]mvinnstr[1]	XSyncValueEqual[1]syncok[1]
DPMSGetVersion[1]attron[1]	XShmPixmapFormat[1]mvinsch[1]	XSyncValueGreaterOrEqual[1]term attrs[1]
DPMSInfo[1]attrset[1]	XShmPutImage[1]mvinsnstr[1]	XSyncValueGreaterThan[1]termna me[1]
DPMSQueryExtension[1]baudrate[1]	XShmQueryExtension[1]mvinsstr[1]	XSyncValueHigh32[1]tgetent[1]
DPMSSetTimeouts[1]beep[1]	XShmQueryVersion[1]mvinstr[1]	XSyncValueIsNegative[1]tgetflag[1]
XSecurityAllocXauth[1]bkgd[1]	XSyncAwait[1]mvprintw[1]	XSyncValueIsPositive[1]tgetnum[1]
XSecurityFreeXauth[1]bkgdset[1]	XSyncChangeAlarm[1]mvscanw[1]	XSyncValueIsZero[1]tgetstr[1]
XSecurityGenerateAuthorization[1]border[1]	XSyncChangeCounter[1]mvvline[1]	XSyncValueLessOrEqual[1]tgoto[1]
XSecurityQueryExtension[1]box[1]	XSyncCreateAlarm[1]mvwaddch[1]	XSyncValueLessThan[1]tgetflag[1]

<code>XSecurityRevokeAuthorization[1]can_change_color[1]</code>	<code>XSyncCreateCounter[1]mvwaddchnstr[1]</code>	<code>XSyncValueLow32[1]tigetnum[1]</code>
<code>XShapeCombineMask[1]cbreak[1]</code>	<code>XSyncDestroyAlarm[1]mvwaddchstr[1]</code>	<code>XSyncValueSubtract[1]tigetstr[1]</code>
<code>XShapeCombineRectangles[1]chgat[1]</code>	<code>XSyncDestroyCounter[1]mvwaddnstr[1]</code>	<code>XdbeAllocateBackBufferName[1]timeout[1]</code>
<code>XShapeCombineRegion[1]clear[1]</code>	<code>XSyncFreeSystemCounterList[1]mvwaddstr[1]</code>	<code>XdbeBeginIdiom[1]touchline[1]</code>
<code>XShapeCombineShape[1]clearok[1]</code>	<code>XSyncGetPriority[1]mvwchgat[1]</code>	<code>XdbeDeallocateBackBufferName[1]touchwin[1]</code>
<code>XShapeGetRectangles[1]clrtoeol[1]</code>	<code>XSyncInitialize[1]mvwdelch[1]</code>	<code>XdbeEndIdiom[1]tparm[1]</code>
<code>XShapeInputSelected[1]clrtoeol[1]</code>	<code>XSyncIntToValue[1]mvwgetch[1]</code>	<code>XdbeFreeVisualInfo[1]tputs[1]</code>
<code>XShapeOffsetShape[1]color_content[1]</code>	<code>XSyncIntsToValue[1]mvwgetnstr[1]</code>	<code>XdbeGetBackBufferAttributes[1]typeahead[1]</code>
<code>XShapeQueryExtension[1]color_set[1]</code>	<code>XSyncListSystemCounters[1]mvwggetstr[1]</code>	<code>XdbeGetVisualInfo[1]unctrl[1]</code>
<code>XShapeQueryExtents[1]copywin[1]</code>	<code>XSyncMaxValue[1]mvwhline[1]</code>	<code>XdbeQueryExtension[1]ungetch[1]</code>
<code>XShapeQueryVersion[1]curs_set[1]</code>	<code>XSyncMinValue[1]mvwin[1]</code>	<code>XdbeSwapBuffers[1]untouchwin[1]</code>
<code>XShapeSelectInput[1]def_prog_mode[1]</code>	<code>XSyncQueryAlarm[1]mvwinch[1]</code>	<code>use_env[1]</code>
<code>XShmAttach[1]def_shell_mode[1]</code>	<code>XSyncQueryCounter[1]mvwinchnstr[1]</code>	<code>vidattr[1]</code>
<code>del_curterm[1]</code>	<code>mvwinchstr[1]</code>	<code>vidputs[1]</code>
<code>delay_output[1]</code>	<code>mvwinnstr[1]</code>	<code>vline[1]</code>
<code>delch[1]</code>	<code>mvwinsch[1]</code>	<code>vw_printw[1]</code>
<code>deleteln[1]</code>	<code>mvwinsnstr[1]</code>	<code>vw_scanw[1]</code>
<code>delscreen[1]</code>	<code>mvwinsstr[1]</code>	<code>vwprintw[1]</code>
<code>delwin[1]</code>	<code>mvwinstr[1]</code>	<code>vwscanw[1]</code>
<code>derwin[1]</code>	<code>mvwprintw[1]</code>	<code>waddch[1]</code>
<code>doupdate[1]</code>	<code>mvwscanw[1]</code>	<code>waddchnstr[1]</code>
<code>dupwin[1]</code>	<code>mvwvline[1]</code>	<code>waddchstr[1]</code>
<code>echo[1]</code>	<code>napms[1]</code>	<code>waddnstr[1]</code>
<code>echochar[1]</code>	<code>newpad[1]</code>	<code>waddstr[1]</code>
<code>endwin[1]</code>	<code>newterm[1]</code>	<code>wattr_get[1]</code>

erase[1]	newwin[1]	wattr_off[1]
erasechar[1]	nl[1]	wattr_on[1]
filter[1]	nocbreak[1]	wattr_set[1]
flash[1]	nodelay[1]	wattroff[1]
flushinp[1]	noecho[1]	wattron[1]
getbkgd[1]	nonl[1]	wattrset[1]
getch[1]	noqiflush[1]	wbkgd[1]
getnstr[1]	noraw[1]	wbkgdset[1]
getstr[1]	notimeout[1]	wborder[1]
getwin[1]	overlay[1]	wchgat[1]
halfdelay[1]	overwrite[1]	wclear[1]
has_colors[1]	pair_content[1]	wclrtoBOT[1]
has_ic[1]	pechochar[1]	wclrtoeol[1]
has_il[1]	pnoutrefresh[1]	wcolor_set[1]
hline[1]	prefresh[1]	wcursyncup[1]
idcok[1]	printw[1]	wdelch[1]
idlok[1]	putp[1]	wdeleteln[1]
immedok[1]	putwin[1]	wechochar[1]
inch[1]	qiflush[1]	werase[1]
inchnstr[1]	raw[1]	wgetch[1]
inchstr[1]	redrawwin[1]	wgetnstr[1]
init_color[1]	refresh[1]	wgetstr[1]
init_pair[1]	reset_prog_mode[1]	whline[1]
initscr[1]	reset_shell_mode[1]	winch[1]
innstr[1]	resetty[1]	winchnstr[1]
insch[1]	restartterm[1]	winchstr[1]
insdelln[1]	riPoffline[1]	winnstr[1]
insertln[1]	savetty[1]	winsch[1]
insnstr[1]	scanw[1]	winsdelln[1]
insstr[1]	scr_dump[1]	winsertln[1]
instr[1]	scr_init[1]	winsnstr[1]

intrflush[1]	scr_restore[1]	winsstr[1]
is_linetouched[1]	scr_set[1]	winstr[1]
is_wintouched[1]	screl[1]	wmove[1]
isendwin[1]	scroll[1]	wnoutrefresh[1]
keyname[1]	scrollok[1]	wprintw[1]
keypad[1]	set_curterm[1]	wredrawln[1]
killchar[1]	set_term[1]	wrefresh[1]
leaveok[1]	setscreg[1]	wscanw[1]
longname[1]	setupterm[1]	wscrl[1]
meta[1]	slk_attr_set[1]	wsetscreg[1]
move[1]	slk_attron[1]	wstandend[1]
mvaddch[1]	slk_attron[1]	wstandout[1]
mvaddchnstr[1]	slk_attrset[1]	wsyncdown[1]
mvaddchstr[1]	slk_clear[1]	wsyncup[1]
mvaddnstr[1]	slk_color[1]	wtimeout[1]
mvaddstr[1]	slk_init[1]	wtouchln[1]
mvchgat[1]	slk_label[1]	wvline[1]
mvcur[1]	slk_noutrefresh[1]	

Table A-8. libncurses Data Interfaces

<u>COLORS</u> <u>ID STD 46 SUS 46</u> <u>CURSES</u>	<u>LINES</u> <u>ID STD 46 SUS 46</u> <u>CURSES</u>	<u>curscr</u> <u>ID STD 46 SUS 46 C</u> <u>URSES</u>
<u>COLOR_PAIRS</u> <u>ID STD 46 S</u> <u>US 46 CURSES</u>	<u>acs_map</u> <u>ID STD 46 SUS 46</u> <u>CURSES</u>	<u>stdscr</u> <u>ID STD 46 SUS 46 C</u> <u>URSES</u>
<u>COLS</u> <u>ID STD 46 SUS 46 C</u> <u>URSES</u>	<u>cur_term</u> <u>ID STD 46 SUS 46</u> <u>CURSES</u>	

A.6. libICElibpam

The behaviour of the interfaces in this library is specified by the following Standards.

this specification

Table A-9. libICElibpam Function Interfaces

<u>IceAcceptConnection</u> <u>[1]pam_acct_</u>	<u>IceGetConnectionContext</u> <u>[1]pam_f</u>	<u>IceProtocolVersion</u> <u>[1]pam_setcred[</u>
--	--	--

mgmt[1]	ail_delay[1]	1]
IceAddConnectionWatch[1]pam_authenticate[1]	IceGetInBufSize[1]pam_get_item[1]	IceReadAuthFileEntry[1]pam_start[1]
IceAllocScratch[1]pam_chauthtok[1]	IceGetListenConnectionNumber[1]pam_getenvlist[1]	IceRegisterForProtocolReply[1]pam_strerror[1]
IceAppLockConn[1]pam_close_session[1]	IceGetListenConnectionString[1]pam_open_session[1]	IceRegisterForProtocolSetup[1]
IceAppUnlockConn[1]pam_end[1]	IceGetOutBufSize[1]pam_set_item[1]	IceRelease[1]
IceAuthFileName[1]	IceInitThreads[1]	IceRemoveConnectionWatch[1]
IceCheckShutdownNegotiation[1]	IceLastReceivedSequenceNumber[1]	IceSetErrorHandler[1]
IceCloseConnection[1]	IceLastSentSequenceNumber[1]	IceSetHostBasedAuthProc[1]

A.7. libpthread

The behaviour of the interfaces in this library is specified by the following Standards.

Large File Support
this specification
ISO POSIX (2003)

Table A-10. libpthread Function Interfaces

IceComposeNetworkIdList[1]pthread_cleanup_pop[1]	IceListenForConnections[1]pthread_create()[1]	IceSetIOErrorHandler[1]pthread_rwlock_trywrlock()[1]
IceConnectionNumber[1]pthread_cleanup_push[1]	IceListenForWellKnownConnections[1]pthread_detach()[1]	IceSetPaAuthData[1]pthread_rwlock_unlock()[1]
IceConnectionStatus[1]pread(GLIBC_2.1)[1]	IceLockAuthFile[1]pthread_equal(GLIBC_2.1)[1]	IceSetShutdownNegotiation[1]pthread_rwlock_wrlock(GLIBC_2.1)[1]
IceConnectionString[1]pread64(GLIBC_2.1)[1]	IceOpenConnection[1]pthread_exit(GLIBC_2.1)[1]	IceSwapping[1]pthread_rwlockattr_destroy(GLIBC_2.1)[1]
IceFlush[1]pthread_attr_destroy(GLIBC_2.0)[1]	IcePing[1]pthread_getspecific(GLIBC_2.0)[1]	IceUnlockAuthFile[1]pthread_rwlockattr_getpshared(GLIBC_2.0)[1]
IceFreeAuthFileEntry[1]pthread_attr_getdetachstate(GLIBC_2.0)[1]	IceProcessMessages[1]pthread_join(GLIBC_2.0)[1]	IceVendor[1]pthread_rwlockattr_init(GLIBC_2.0)[1]
IceFreeListenObjs[1]pthread_attr_getguardsize(GLIBC_2.1)[1]	IceProtocolRevision[1]pthread_key_create(GLIBC_2.1)[1]	IceWriteAuthFileEntry[1]pthread_rwlockattr_setpshared(GLIBC_2.1)[1]
IceGenerateMagicCookie[1]pthread	IceProtocolSetup[1]pthread_key_de	pthread_self(GLIBC_2.0)[1]

<code>_attr_getschedparam(GLIBC_2.0)[1]</code>	<code>lete(GLIBC_2.0)[1]</code>	
<code>IceGetAuthFileEntry[1]pthread_attr_getstackaddr(GLIBC_2.1)[1]</code>	<code>IceProtocolShutdown[1]pthread_kill(GLIBC_2.1)[1]</code>	<code>pthread_setcancelstate(GLIBC_2.1)[1]</code>

A.7. libSM

The behaviour of the interfaces in this library is specified by the following Standards.

Table A-10. libSM Function Interfaces

<code>pthread_attr_getstacksize(GLIBC_2.1)[1]</code>	<code>pthread_mutex_destroy(GLIBC_2.1)[1]</code>	<code>pthread_setcanceltype(GLIBC_2.1)[1]</code>
<code>SmFreeProperty[1]pthread_attr_init(GLIBC_2.1)[1]</code>	<code>SmcRelease[1]pthread_mutex_init(GLIBC_2.1)[1]</code>	<code>SmsInitialize[1]pthread_setconcurrency[1]</code>
<code>SmFreeReasons[1]pthread_attr_setdetachstate(GLIBC_2.0)[1]</code>	<code>SmcRequestSaveYourself[1]pthread_mutex_lock(GLIBC_2.0)[1]</code>	<code>SmsInteract[1]pthread_setspecific(GLIBC_2.0)[1]</code>
<code>SmcClientID[1]pthread_attr_setguardsize(GLIBC_2.1)[1]</code>	<code>SmcRequestSaveYourselfPhase2[1]pthread_mutex_trylock(GLIBC_2.1)[1]</code>	<code>SmsProtocolRevision[1]pthread_sigmask(GLIBC_2.1)[1]</code>
<code>SmcCloseConnection[1]pthread_attr_setschedparam(GLIBC_2.0)[1]</code>	<code>SmcSaveYourselfDone[1]pthread_mutex_unlock(GLIBC_2.0)[1]</code>	<code>SmsProtocolVersion[1]pthread_testcancel(GLIBC_2.0)[1]</code>
<code>SmcDeleteProperties[1]pthread_attr_setstackaddr(GLIBC_2.1)[1]</code>	<code>SmcSetErrorHandler[1]pthread_mutexattr_destroy(GLIBC_2.1)[1]</code>	<code>SmsRegisterClientReply[1]pwrite(GLIBC_2.1)[1]</code>
<code>SmcGetIdleConnection[1]pthread_attr_setstacksize(GLIBC_2.1)[1]</code>	<code>SmcSetProperties[1]pthread_mutexattr_getpshared(GLIBC_2.1)[1]</code>	<code>SmsReturnProperties[1]pwrite64(GLIBC_2.1)[1]</code>
<code>SmcGetProperties[1]pthread_cancel(GLIBC_2.0)[1]</code>	<code>SmcVendor[1]pthread_mutexattr_gettype(GLIBC_2.0)[1]</code>	<code>SmsSaveComplete[1]sem_close(GLIBC_2.0)[1]</code>
<code>SmcInteractDone[1]pthread_cond_broadcast(GLIBC_2.0)[1]</code>	<code>SmsCleanUp[1]pthread_mutexattr_init(GLIBC_2.0)[1]</code>	<code>SmsSaveYourself[1]sem_destroy(GLIBC_2.0)[1]</code>
<code>SmcInteractRequest[1]pthread_cond_destroy(GLIBC_2.0)[1]</code>	<code>SmsClientHostName[1]pthread_mutexattr_setpshared(GLIBC_2.0)[1]</code>	<code>SmsSaveYourselfPhase2[1]sem_getvalue(GLIBC_2.0)[1]</code>
<code>SmcModifyCallbacks[1]pthread_cond_init(GLIBC_2.0)[1]</code>	<code>SmsClientID[1]pthread_mutexattr_settype(GLIBC_2.0)[1]</code>	<code>SmsSetErrorHandler[1]sem_init(GLIBC_2.0)[1]</code>
<code>SmcOpenConnection[1]pthread_cond_signal(GLIBC_2.0)[1]</code>	<code>SmsDie[1]pthread_once(GLIBC_2.0)[1]</code>	<code>SmsShutdownCancelled[1]sem_open(GLIBC_2.0)[1]</code>
<code>SmcProtocolRevision[1]pthread_co</code>	<code>SmsGenerateClientID[1]pthread_r</code>	<code>sem_post(GLIBC_2.0)[1]</code>

nd_timedwait(GLIBC_2.0)[1]	wlock_destroy(GLIBC_2.0)[1]	
SmeProtocolVersion[1] pthread_cond_wait(GLIBC_2.0)[1]	SmsGetIceConnection[1] pthread_rwlock_init(GLIBC_2.0)[1]	sem_timedwait(GLIBC_2.0)[1]
pthread_condattr_destroy(GLIBC_2.0)[1]	pthread_rwlock_rdlock(GLIBC_2.0)[1]	sem_trywait(GLIBC_2.0)[1]
pthread_condattr_getpshared[1]	pthread_rwlock_timedrdlock[1]	sem_unlink()[1]
pthread_condattr_init(GLIBC_2.0)[1]	pthread_rwlock_timedwrlock[1]	sem_wait(GLIBC_2.0)[1]
pthread_condattr_setpshared[1]	pthread_rwlock_tryrdlock()[1]	

A.8. ~~libdl~~libutil

The behaviour of the interfaces in this library is specified by the following Standards.

~~Linux Standard Base~~this specification

~~ISO/IEC 9945:2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3~~

Table A-11. ~~libdl~~ Function Interfaces

Table A-11. libutil Function Interfaces

forkpty(GLIBC_2.0)[1]	login_tty(GLIBC_2.0)[1]	logwtmp(GLIBC_2.0)[1]
dladdr(GLIBC_2.0) login(GLIBC_2.0)[1]	derror(GLIBC_2.0) logout(GLIBC_2.0)[1]	dlsym(GLIBC_2.0) openpty(GLIBC_2.0)[1]
dlclose(GLIBC_2.0)[1]	dlopen(GLIBC_2.0)[1]	

A.9. ~~libcrypt~~libz

The behaviour of the interfaces in this library is specified by the following Standards.

~~ISO/IEC 9945:2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3~~zlib Manual

Table A-12. ~~libcrypt~~libz Function Interfaces

crypt(GLIBC_2.0) adler32[1]	encrypt(GLIBC_2.0) gzdopen[1]	setkey(GLIBC_2.0) gztell[1]
--	--	--

A.10. libz

The behaviour of the interfaces in this library is specified by the following Standards.

zlib 1.2 Manual

Table A-13. libz Function Interfaces

compress[1]	gzEOF[1]	gzwrite[1]
-------------	----------	------------

adler32compress2[1]	gzdopengzerror[1]	gztellinflate[1]
compresscrc32[1]	gzEOFgzflush[1]	gzwriteinflateEnd[1]
compress2deflate[1]	gzerrorgzgetc[1]	inflateinflateInit2_[1]
crc32deflateCopy[1]	gzflushgzgets[1]	inflateEndinflateInit_[1]
deflatedeflateEnd[1]	gzgetgzopen[1]	inflateInit2_inflateReset[1]
deflateCopydeflateInit2_[1]	gzgetsgzprintf[1]	inflateInit_inflateSetDictionary[1]
deflateEnddeflateInit_[1]	gzopengzputc[1]	inflateResetinflateSync[1]
deflateInit2_deflateParams[1]	gzprintfgzputs[1]	inflateSetDictionaryinflateSyncPoint[1]
deflateInit_deflateReset[1]	gzputgzread[1]	inflateSyncuncompress[1]
deflateParamsdeflateSetDictionary[1]	gzputsgzrewind[1]	inflateSyncPointzError[1]
deflateResetget_crc_table[1]	gzreadgzseek[1]	uncompress[1]
deflateSetDictionarygzclose[1]	gzrewindgzsetparams[1]	zError[1]
get_crc_table[1]	gzseek[1]	
gzclose[1]	gzsetparams[1]	

A.11. libncurses

The behaviour of the interfaces in this library is specified by the following Standards:

CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1 85912 171 3, C610), plus Corrigendum U018

Table A-14. libncurses Function Interfaces

addch[1]	mvdele[1]	slk_refresh[1]
addchnstr[1]	mvderwin[1]	slk_restore[1]
addchstr[1]	mvgetch[1]	slk_set[1]
addnstr[1]	mvgetnstr[1]	slk_touch[1]
addstr[1]	mvgetstr[1]	standend[1]
attr_get[1]	mvhline[1]	standout[1]
attr_off[1]	mvinch[1]	start_color[1]
attr_on[1]	mvinchnstr[1]	subpad[1]
attr_set[1]	mvinchstr[1]	subwin[1]

attroff[1]	mvinnstr[1]	syneok[1]
attron[1]	mvinsch[1]	termattrs[1]
attrset[1]	mvinsnstr[1]	termname[1]
baudrate[1]	mvinsstr[1]	tgetent[1]
beep[1]	mvinstr[1]	tgetflag[1]
bkgd[1]	mvprintw[1]	tgetnum[1]
bkgdset[1]	mvscanw[1]	tgetstr[1]
border[1]	mvvline[1]	tgoto[1]
box[1]	mvwaddeh[1]	tigetflag[1]
can_change_color[1]	mvwaddehnstr[1]	tigetnum[1]
cbreak[1]	mvwaddehstr[1]	tigetstr[1]
chgat[1]	mvwaddnstr[1]	timeout[1]
clear[1]	mvwaddstr[1]	touchline[1]
clearok[1]	mvwehgat[1]	touchwin[1]
clrtobot[1]	mvwdeleh[1]	tparm[1]
clrtoeol[1]	mvwgeteh[1]	tputs[1]
color_content[1]	mvwgetnstr[1]	typeahead[1]
color_set[1]	mvwgetstr[1]	unctrl[1]
copywin[1]	mvwhline[1]	ungetch[1]
curs_set[1]	mvwin[1]	untouchwin[1]
def_prog_mode[1]	mvwinch[1]	use_env[1]
def_shell_mode[1]	mvwinchnstr[1]	vidattr[1]
del_curterm[1]	mvwinchstr[1]	vidputs[1]
delay_output[1]	mvwinnstr[1]	vline[1]
deleh[1]	mvwinsch[1]	vw_printw[1]
deleteln[1]	mvwinsnstr[1]	vw_scanw[1]
delscreen[1]	mvwinsstr[1]	vwprintw[1]
delwin[1]	mvwinstr[1]	vwscanw[1]
derwin[1]	mvwprintw[1]	waddeh[1]
doupdate[1]	mvwscanw[1]	waddehnstr[1]
dupwin[1]	mvwvline[1]	waddehstr[1]

echo[1]	napms[1]	waddnstr[1]
echochar[1]	newpad[1]	waddstr[1]
endwin[1]	newterm[1]	wattr_get[1]
erase[1]	newwin[1]	wattr_off[1]
erasechar[1]	nl[1]	wattr_on[1]
filter[1]	noebreak[1]	wattr_set[1]
flash[1]	nodelay[1]	wattroff[1]
flushinp[1]	noecho[1]	wattron[1]
getbkgd[1]	nonl[1]	wattrset[1]
getch[1]	noqiflush[1]	wbkgd[1]
getnstr[1]	noraw[1]	wbkgdset[1]
getstr[1]	notimeout[1]	wborder[1]
getwin[1]	overlay[1]	wchgat[1]
halfdelay[1]	overwrite[1]	wclear[1]
has_colors[1]	pair_content[1]	welrtobot[1]
has_ic[1]	pechochar[1]	welrtoeol[1]
has_il[1]	pnoutrefresh[1]	wecolor_set[1]
hline[1]	prefresh[1]	weursyncup[1]
ideok[1]	printw[1]	wdeleh[1]
idlok[1]	putp[1]	wdeleteln[1]
immedok[1]	putwin[1]	wechochar[1]
inch[1]	qiflush[1]	werase[1]
inehnstr[1]	raw[1]	wgetch[1]
inchstr[1]	redrawwin[1]	wgetnstr[1]
init_color[1]	refresh[1]	wgetstr[1]
init_pair[1]	reset_prog_mode[1]	whline[1]
initser[1]	reset_shell_mode[1]	winch[1]
innstr[1]	resetty[1]	winehnstr[1]
inseh[1]	restartterm[1]	winchstr[1]
insdelln[1]	ripline[1]	winnstr[1]
insertln[1]	savetty[1]	winsch[1]

insnstr[1]	scanw[1]	winsdelln[1]
insstr[1]	ser_dump[1]	winsertln[1]
instr[1]	ser_init[1]	winsnstr[1]
intrflush[1]	ser_restore[1]	winsstr[1]
is_linetouched[1]	ser_set[1]	winstr[1]
is_wintouched[1]	serl[1]	wmove[1]
isendwin[1]	scroll[1]	wnoutrefresh[1]
keyname[1]	serollok[1]	wprintw[1]
keypad[1]	set_curterm[1]	wredrawln[1]
killchar[1]	set_term[1]	wrefresh[1]
leaveok[1]	setserreg[1]	wscanw[1]
longname[1]	setupterm[1]	wserl[1]
meta[1]	slk_attr_set[1]	wsetserreg[1]
move[1]	slk_attroff[1]	wstandend[1]
mvaddeh[1]	slk_attron[1]	wstandout[1]
mvaddehnstr[1]	slk_attrset[1]	wsynedown[1]
mvaddehstr[1]	slk_clear[1]	wsyneup[1]
mvaddnstr[1]	slk_color[1]	wtimeout[1]
mvaddstr[1]	slk_init[1]	wtouchln[1]
mvchgat[1]	slk_label[1]	wvline[1]
mvcur[1]	slk_noutrefresh[1]	

Table A-15. libncurses Data Interfaces

COLORS	LINES	curser
COLOR_PAIRS	acs_map	stdscr
COLS	cur_term	

A.12. libutil

The behaviour of the interfaces in this library is specified by the following Standards.

Linux Standard Base

Table A-16. libutil Function Interfaces

forkpty(GLIBC_2.0)[1]	login_tty(GLIBC_2.0)[1]	logwtmp(GLIBC_2.0)[1]
login(GLIBC_2.0)[1]	logout(GLIBC_2.0)[1]	openpty(GLIBC_2.0)[1]

A.13. libe

The behaviour of the interfaces in this library is specified by the following Standards.

ISO/IEC 9899: 1999, Programming Languages—C

Large File Support

Linux Standard Base

CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0, C606)

ISO/IEC 9945:2003 Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3

System V Interface Definition, Issue 3 (ISBN 0201566524)

System V Interface Definition, Fourth Edition

Table A-17. libe Function Interfaces

_Exit(GLIBC_2.1.1)[1]	getrusage(GLIBC_2.1.1)[1]	sigaddset(GLIBC_2.1.1)[1]
_IO_feof(GLIBC_2.0)[1]	getservbyname(GLIBC_2.0)[1]	sigaltstack(GLIBC_2.0)[1]
_IO_getc(GLIBC_2.0)[1]	getservbyport(GLIBC_2.0)[1]	sigandset(GLIBC_2.0)[1]
_IO_pute(GLIBC_2.0)[1]	getservent(GLIBC_2.0)[1]	sigblock(GLIBC_2.0)[1]
_IO_puts(GLIBC_2.0)[1]	getsid(GLIBC_2.0)[1]	sigdelset(GLIBC_2.0)[1]
__assert_fail(GLIBC_2.0)[1]	getsockname(GLIBC_2.0)[1]	sigemptyset(GLIBC_2.0)[1]
__ctype_b_loc[1]	getsockopt[1]	sigfillset[1]
__ctype_get_mb_cur_max(GLIBC_2.0)[1]	getsubopt(GLIBC_2.0)[1]	siggetmask(GLIBC_2.0)[1]
__ctype_tolower_loc[1]	gettext[1]	sighold[1]
__ctype_toupper_loc[1]	gettimeofday[1]	sigignore[1]
__exa_atexit(GLIBC_2.1.3)[1]	getuid(GLIBC_2.1.3)[1]	siginterrupt(GLIBC_2.1.3)[1]
__errno_location(GLIBC_2.0)[1]	getutent(GLIBC_2.0)[1]	sigisemptyset(GLIBC_2.0)[1]
__fpending(GLIBC_2.2)[1]	getutent_r(GLIBC_2.2)[1]	sigismember(GLIBC_2.2)[1]
__fxstat(GLIBC_2.0)[1]	getutxent(GLIBC_2.0)[1]	siglongjmp(GLIBC_2.0)[1]
__fxstat64(GLIBC_2.2)[1]	getutxid(GLIBC_2.2)[1]	signal(GLIBC_2.2)[1]
__getpagesize(GLIBC_2.0)[1]	getutxline(GLIBC_2.0)[1]	sigorset(GLIBC_2.0)[1]
__getpgid(GLIBC_2.0)[1]	getw(GLIBC_2.0)[1]	sigpause(GLIBC_2.0)[1]

<code>__h_errno_location[1]</code>	<code>getwc()[1]</code>	<code>sigpending()[1]</code>
<code>__isinf[1]</code>	<code>getwchar()[1]</code>	<code>sigprocmask()[1]</code>
<code>__isnff[1]</code>	<code>getwd()[1]</code>	<code>sigqueue()[1]</code>
<code>__isnfl[1]</code>	<code>glob()[1]</code>	<code>sigrelse()[1]</code>
<code>__isnan[1]</code>	<code>glob64()[1]</code>	<code>sigreturn()[1]</code>
<code>__isnanf[1]</code>	<code>globfree()[1]</code>	<code>sigset()[1]</code>
<code>__isnaml[1]</code>	<code>globfree64()[1]</code>	<code>sigstack()[1]</code>
<code>__libc_current_sigrtmax(GLIBC_2.1)[1]</code>	<code>gmtime(GLIBC_2.1)[1]</code>	<code>sigsuspend(GLIBC_2.1)[1]</code>
<code>__libc_current_sigrtmin(GLIBC_2.1)[1]</code>	<code>gmtime_r(GLIBC_2.1)[1]</code>	<code>sigtimedwait(GLIBC_2.1)[1]</code>
<code>__libc_start_main(GLIBC_2.0)[1]</code>	<code>grantpt(GLIBC_2.0)[1]</code>	<code>sigwait(GLIBC_2.0)[1]</code>
<code>__lxstat(GLIBC_2.0)[1]</code>	<code>hereate(GLIBC_2.0)[1]</code>	<code>sigwaitinfo(GLIBC_2.0)[1]</code>
<code>__lxstat64(GLIBC_2.2)[1]</code>	<code>hdestroy(GLIBC_2.2)[1]</code>	<code>sleep(GLIBC_2.2)[1]</code>
<code>__mempepy(GLIBC_2.0)[1]</code>	<code>hsearch(GLIBC_2.0)[1]</code>	<code>snprintf(GLIBC_2.0)[1]</code>
<code>__rawmemchr(GLIBC_2.1)[1]</code>	<code>htonl(GLIBC_2.1)[1]</code>	<code>socket(GLIBC_2.1)[1]</code>
<code>__register_atfork[1]</code>	<code>htons()[1]</code>	<code>socketpair()[1]</code>
<code>__sigsetjmp(GLIBC_2.0)[1]</code>	<code>iconv(GLIBC_2.0)[1]</code>	<code>sprintf(GLIBC_2.0)[1]</code>
<code>__stpepy(GLIBC_2.0)[1]</code>	<code>iconv_close(GLIBC_2.0)[1]</code>	<code>srand(GLIBC_2.0)[1]</code>
<code>__strdup(GLIBC_2.0)[1]</code>	<code>iconv_open(GLIBC_2.0)[1]</code>	<code>srand48(GLIBC_2.0)[1]</code>
<code>__strtod_internal(GLIBC_2.0)[1]</code>	<code>imaxabs(GLIBC_2.0)[1]</code>	<code>srandom(GLIBC_2.0)[1]</code>
<code>__strtof_internal(GLIBC_2.0)[1]</code>	<code>imaxdiv(GLIBC_2.0)[1]</code>	<code>sscanf(GLIBC_2.0)[1]</code>
<code>__strtok_r(GLIBC_2.0)[1]</code>	<code>index(GLIBC_2.0)[1]</code>	<code>statvfs(GLIBC_2.0)[1]</code>
<code>__strtol_internal(GLIBC_2.0)[1]</code>	<code>inet_addr(GLIBC_2.0)[1]</code>	<code>statvfs64[1]</code>
<code>__strtold_internal(GLIBC_2.0)[1]</code>	<code>inet_ntoa(GLIBC_2.0)[1]</code>	<code>stime(GLIBC_2.0)[1]</code>
<code>__strtol_internal(GLIBC_2.0)[1]</code>	<code>inet_ntop[1]</code>	<code>stpepy(GLIBC_2.0)[1]</code>
<code>__strtoul_internal(GLIBC_2.0)[1]</code>	<code>inet_pton[1]</code>	<code>stpncpy(GLIBC_2.0)[1]</code>
<code>__strtoull_internal(GLIBC_2.0)[1]</code>	<code>initgroups(GLIBC_2.0)[1]</code>	<code>streaseemp(GLIBC_2.0)[1]</code>
<code>__sysconf(GLIBC_2.2)[1]</code>	<code>initstate(GLIBC_2.2)[1]</code>	<code>streasestr(GLIBC_2.2)[1]</code>
<code>__sysv_signal(GLIBC_2.0)[1]</code>	<code>insque(GLIBC_2.0)[1]</code>	<code>streat(GLIBC_2.0)[1]</code>
<code>__westod_internal(GLIBC_2.0)[1]</code>	<code>ioctl(GLIBC_2.0)[1]</code>	<code>strechr(GLIBC_2.0)[1]</code>

__westof_internal(GLIBC_2.0)[1]	isalnum(GLIBC_2.0)[1]	strempt(GLIBC_2.0)[1]
__westol_internal(GLIBC_2.0)[1]	isalpha(GLIBC_2.0)[1]	streoll(GLIBC_2.0)[1]
__westold_internal(GLIBC_2.0)[1]	isascii(GLIBC_2.0)[1]	strep(GLIBC_2.0)[1]
__westoul_internal(GLIBC_2.0)[1]	isatty(GLIBC_2.0)[1]	strespn(GLIBC_2.0)[1]
__xmknod(GLIBC_2.0)[1]	isblank(GLIBC_2.0)[1]	strdup(GLIBC_2.0)[1]
__xstat(GLIBC_2.0)[1]	iscentrl(GLIBC_2.0)[1]	strerror(GLIBC_2.0)[1]
__xstat64(GLIBC_2.2)[1]	isdigit(GLIBC_2.2)[1]	strerror_r(GLIBC_2.2)[1]
_exit(GLIBC_2.0)[1]	isgraph(GLIBC_2.0)[1]	strfmon(GLIBC_2.0)[1]
_longjmp(GLIBC_2.0)[1]	isinf(GLIBC_2.0)[1]	strfry(GLIBC_2.0)[1]
_obstack_begin(GLIBC_2.0)[1]	isinf[1]	strftime(GLIBC_2.0)[1]
_obstack_newchunk(GLIBC_2.0)[1]	isinf(GLIBC_2.0)[1]	strlen(GLIBC_2.0)[1]
_setjmp(GLIBC_2.0)[1]	islower(GLIBC_2.0)[1]	strncasecmp(GLIBC_2.0)[1]
_tolower(GLIBC_2.0)[1]	isnan(GLIBC_2.0)[1]	strncat(GLIBC_2.0)[1]
_toupper(GLIBC_2.0)[1]	isnanf(GLIBC_2.0)[1]	strncmp(GLIBC_2.0)[1]
a64l(GLIBC_2.0)[1]	isnanl(GLIBC_2.0)[1]	strncpy(GLIBC_2.0)[1]
abort(GLIBC_2.0)[1]	isprint(GLIBC_2.0)[1]	strndup(GLIBC_2.0)[1]
abs(GLIBC_2.0)[1]	ispunct(GLIBC_2.0)[1]	strnlen(GLIBC_2.0)[1]
accept(GLIBC_2.0)[1]	isspace(GLIBC_2.0)[1]	strpbrk(GLIBC_2.0)[1]
access(GLIBC_2.0)[1]	isupper(GLIBC_2.0)[1]	strptime(GLIBC_2.0)[1]
acct(GLIBC_2.0)[1]	iswalnum(GLIBC_2.0)[1]	strchr(GLIBC_2.0)[1]
adjtime(GLIBC_2.0)[1]	iswalpha(GLIBC_2.0)[1]	strsep(GLIBC_2.0)[1]
alarm(GLIBC_2.0)[1]	iswblank(GLIBC_2.0)[1]	strsignal(GLIBC_2.0)[1]
asctime(GLIBC_2.0)[1]	iswcentrl(GLIBC_2.0)[1]	strspn(GLIBC_2.0)[1]
asctime_r(GLIBC_2.0)[1]	iswctype(GLIBC_2.0)[1]	strstr(GLIBC_2.0)[1]
asprintf(GLIBC_2.0)[1]	iswdigit(GLIBC_2.0)[1]	strtod(GLIBC_2.0)[1]
atof(GLIBC_2.0)[1]	iswgraph(GLIBC_2.0)[1]	strtof(GLIBC_2.0)[1]
atoi(GLIBC_2.0)[1]	iswlower(GLIBC_2.0)[1]	strtoimax(GLIBC_2.0)[1]
atol(GLIBC_2.0)[1]	iswprint(GLIBC_2.0)[1]	strtok(GLIBC_2.0)[1]
atoll[1]	iswpunct()[1]	strtok_r[1]
authnone_create(GLIBC_2.0)[1]	iswspace(GLIBC_2.0)[1]	strtol(GLIBC_2.0)[1]

basename(GLIBC_2.0)[1]	iswupper(GLIBC_2.0)[1]	strtold(GLIBC_2.0)[1]
bcmp(GLIBC_2.0)[1]	iswxdigit(GLIBC_2.0)[1]	strtoll(GLIBC_2.0)[1]
bcopy(GLIBC_2.0)[1]	isxdigit(GLIBC_2.0)[1]	strtoq(GLIBC_2.0)[1]
bind(GLIBC_2.0)[1]	jrand48(GLIBC_2.0)[1]	strtoul(GLIBC_2.0)[1]
bind_textdomain_codeset[1]	key_decryptsession()[1]	strtoull()[1]
bindresvport(GLIBC_2.0)[1]	kill(GLIBC_2.0)[1]	strtoumax(GLIBC_2.0)[1]
bindtextdomain(GLIBC_2.0)[1]	killpg(GLIBC_2.0)[1]	strtouq(GLIBC_2.0)[1]
brk(GLIBC_2.0)[1]	l64a(GLIBC_2.0)[1]	strversemp(GLIBC_2.0)[1]
bsd_signal(GLIBC_2.0)[1]	labs(GLIBC_2.0)[1]	strxfrm(GLIBC_2.0)[1]
bsearch(GLIBC_2.0)[1]	lchown(GLIBC_2.0)[1]	sve_getreqset(GLIBC_2.0)[1]
btowc(GLIBC_2.0)[1]	leong48(GLIBC_2.0)[1]	sve_register(GLIBC_2.0)[1]
bzero(GLIBC_2.0)[1]	ldiv(GLIBC_2.0)[1]	sve_run(GLIBC_2.0)[1]
calloc(GLIBC_2.0)[1]	lfind(GLIBC_2.0)[1]	sve_sendreply(GLIBC_2.0)[1]
catclose(GLIBC_2.0)[1]	link(GLIBC_2.0)[1]	sveerr_auth(GLIBC_2.0)[1]
catgets(GLIBC_2.0)[1]	listen(GLIBC_2.0)[1]	sveerr_decode(GLIBC_2.0)[1]
catopen(GLIBC_2.0)[1]	llabs(GLIBC_2.0)[1]	sveerr_noproc(GLIBC_2.0)[1]
efgetispeed(GLIBC_2.0)[1]	lldiv(GLIBC_2.0)[1]	sveerr_noprogram(GLIBC_2.0)[1]
efgetospeed(GLIBC_2.0)[1]	localeconv(GLIBC_2.0)[1]	sveerr_progvers(GLIBC_2.0)[1]
efmakeraw(GLIBC_2.0)[1]	localtime(GLIBC_2.0)[1]	sveerr_systemerr(GLIBC_2.0)[1]
efsetispeed(GLIBC_2.0)[1]	localtime_r(GLIBC_2.0)[1]	sveerr_weakauth(GLIBC_2.0)[1]
efsetospeed(GLIBC_2.0)[1]	lockf(GLIBC_2.0)[1]	svetep_create(GLIBC_2.0)[1]
efsetspeed(GLIBC_2.0)[1]	lockf64(GLIBC_2.0)[1]	sveudp_create(GLIBC_2.0)[1]
chdir(GLIBC_2.0)[1]	longjmp(GLIBC_2.0)[1]	swab(GLIBC_2.0)[1]
chmod(GLIBC_2.0)[1]	lrand48(GLIBC_2.0)[1]	swapecontext(GLIBC_2.0)[1]
chown(GLIBC_2.1)[1]	lsearch(GLIBC_2.1)[1]	swprintf(GLIBC_2.1)[1]
chroot(GLIBC_2.0)[1]	lseek(GLIBC_2.0)[1]	swscanf(GLIBC_2.0)[1]
clearerr(GLIBC_2.0)[1]	lseek64(GLIBC_2.0)[1]	symlink(GLIBC_2.0)[1]
clnt_create(GLIBC_2.0)[1]	makecontext(GLIBC_2.0)[1]	syne(GLIBC_2.0)[1]
clnt_pcreateerror(GLIBC_2.0)[1]	malloc(GLIBC_2.0)[1]	sysconf(GLIBC_2.0)[1]
clnt_perrno(GLIBC_2.0)[1]	mblen(GLIBC_2.0)[1]	syslog(GLIBC_2.0)[1]
clnt_perror(GLIBC_2.0)[1]	mbrlen(GLIBC_2.0)[1]	system(GLIBC_2.0)[1]

elnt_spercreateerror(GLIBC_2.0)[1]	mbrtowe(GLIBC_2.0)[1]	tedrain(GLIBC_2.0)[1]
elnt_spereno(GLIBC_2.0)[1]	mbsinit(GLIBC_2.0)[1]	teflow(GLIBC_2.0)[1]
elnt_spererror(GLIBC_2.0)[1]	mbsnrtowes(GLIBC_2.0)[1]	teflush(GLIBC_2.0)[1]
clock(GLIBC_2.0)[1]	mbsrtowes(GLIBC_2.0)[1]	tegetattr(GLIBC_2.0)[1]
close(GLIBC_2.0)[1]	mbstowes(GLIBC_2.0)[1]	tegetpgrp(GLIBC_2.0)[1]
closedir(GLIBC_2.0)[1]	mbtowe(GLIBC_2.0)[1]	tegetsid(GLIBC_2.0)[1]
closelog(GLIBC_2.0)[1]	memecpy(GLIBC_2.0)[1]	tesendbreak(GLIBC_2.0)[1]
confstr(GLIBC_2.0)[1]	memchr(GLIBC_2.0)[1]	tesetattr(GLIBC_2.0)[1]
connect(GLIBC_2.0)[1]	memcmp(GLIBC_2.0)[1]	tesetpgrp(GLIBC_2.0)[1]
creat(GLIBC_2.0)[1]	memcpy(GLIBC_2.0)[1]	tdelete[1]
creat64(GLIBC_2.1)[1]	memmem(GLIBC_2.1)[1]	telldir(GLIBC_2.1)[1]
ctermid(GLIBC_2.0)[1]	memmove(GLIBC_2.0)[1]	tempnam(GLIBC_2.0)[1]
ctime(GLIBC_2.0)[1]	memrchr(GLIBC_2.0)[1]	textdomain(GLIBC_2.0)[1]
ctime_r(GLIBC_2.0)[1]	memset(GLIBC_2.0)[1]	tfind(GLIBC_2.0)[1]
cuserid(GLIBC_2.0)[1]	mkdir(GLIBC_2.0)[1]	time(GLIBC_2.0)[1]
daemon(GLIBC_2.0)[1]	mkfifo(GLIBC_2.0)[1]	times(GLIBC_2.0)[1]
degettext(GLIBC_2.0)[1]	mkstemp(GLIBC_2.0)[1]	tmpfile(GLIBC_2.0)[1]
dengettext[1]	mkstemp64()[1]	tmpfile64()[1]
dgettext[1]	mktemp()[1]	tmpnam()[1]
difftime(GLIBC_2.0)[1]	mktime(GLIBC_2.0)[1]	toascii(GLIBC_2.0)[1]
dirname(GLIBC_2.0)[1]	mlock(GLIBC_2.0)[1]	tolower(GLIBC_2.0)[1]
div(GLIBC_2.0)[1]	mlockall(GLIBC_2.0)[1]	toupper(GLIBC_2.0)[1]
dngettext[1]	mmap()[1]	towetrans()[1]
drand48(GLIBC_2.0)[1]	mmap64(GLIBC_2.0)[1]	tolower(GLIBC_2.0)[1]
dup(GLIBC_2.0)[1]	mprotect(GLIBC_2.0)[1]	toupper(GLIBC_2.0)[1]
dup2(GLIBC_2.0)[1]	mrand48(GLIBC_2.0)[1]	truncate(GLIBC_2.0)[1]
eevt(GLIBC_2.0)[1]	msgctl(GLIBC_2.0)[1]	truncate64(GLIBC_2.0)[1]
endgrent(GLIBC_2.0)[1]	msgget(GLIBC_2.0)[1]	tsearch(GLIBC_2.0)[1]
endnetent(GLIBC_2.0)[1]	msgrev(GLIBC_2.0)[1]	ttynam(GLIBC_2.0)[1]
endprotoent(GLIBC_2.0)[1]	msgsnd(GLIBC_2.0)[1]	ttynam_r(GLIBC_2.0)[1]
endpwent(GLIBC_2.0)[1]	msync(GLIBC_2.0)[1]	twalk(GLIBC_2.0)[1]

endservent(GLIBC_2.0)[1]	munlock(GLIBC_2.0)[1]	tzset(GLIBC_2.0)[1]
endutent(GLIBC_2.0)[1]	munlockall(GLIBC_2.0)[1]	ualarm(GLIBC_2.0)[1]
endutxent(GLIBC_2.1)[1]	munmap(GLIBC_2.1)[1]	ulimit(GLIBC_2.1)[1]
erand48(GLIBC_2.0)[1]	nanosleep(GLIBC_2.0)[1]	umask(GLIBC_2.0)[1]
err(GLIBC_2.0)[1]	nftw(GLIBC_2.0)[1]	uname(GLIBC_2.0)[1]
error(GLIBC_2.0)[1]	nftw64(GLIBC_2.0)[1]	ungetc(GLIBC_2.0)[1]
errx(GLIBC_2.0)[1]	ngettext[1]	ungetwe(GLIBC_2.0)[1]
execl(GLIBC_2.0)[1]	nice(GLIBC_2.0)[1]	unlink(GLIBC_2.0)[1]
execle(GLIBC_2.0)[1]	nl_langinfo(GLIBC_2.0)[1]	unlockpt(GLIBC_2.0)[1]
execlp(GLIBC_2.0)[1]	nrnd48(GLIBC_2.0)[1]	unsetenv[1]
execv(GLIBC_2.0)[1]	ntohl(GLIBC_2.0)[1]	usleep(GLIBC_2.0)[1]
execve(GLIBC_2.0)[1]	ntohs(GLIBC_2.0)[1]	utime(GLIBC_2.0)[1]
execvp(GLIBC_2.0)[1]	obstack_free(GLIBC_2.0)[1]	utimes(GLIBC_2.0)[1]
exit(GLIBC_2.0)[1]	open(GLIBC_2.0)[1]	vasprintf(GLIBC_2.0)[1]
fehdir(GLIBC_2.0)[1]	open64(GLIBC_2.0)[1]	vdprintf(GLIBC_2.0)[1]
fehmod(GLIBC_2.0)[1]	opendir(GLIBC_2.0)[1]	verrx(GLIBC_2.0)[1]
fehown(GLIBC_2.0)[1]	openlog(GLIBC_2.0)[1]	vfork(GLIBC_2.0)[1]
felose(GLIBC_2.1)[1]	patheconf(GLIBC_2.1)[1]	vfprintf(GLIBC_2.1)[1]
fentl(GLIBC_2.0)[1]	pause(GLIBC_2.0)[1]	vfscanf[1]
fevt(GLIBC_2.0)[1]	pelose(GLIBC_2.0)[1]	vwprintf(GLIBC_2.0)[1]
fdatasync(GLIBC_2.0)[1]	perror(GLIBC_2.0)[1]	vfwscanf(GLIBC_2.0)[1]
fdopen(GLIBC_2.1)[1]	pipe(GLIBC_2.1)[1]	vprintf(GLIBC_2.1)[1]
feof(GLIBC_2.0)[1]	pmap_getport(GLIBC_2.0)[1]	vscanf[1]
ferror(GLIBC_2.0)[1]	pmap_set(GLIBC_2.0)[1]	vsnprintf(GLIBC_2.0)[1]
fflush(GLIBC_2.0)[1]	pmap_unset(GLIBC_2.0)[1]	vsprintf(GLIBC_2.0)[1]
fflush_unlocked(GLIBC_2.0)[1]	poll(GLIBC_2.0)[1]	vsscanf[1]
ffs(GLIBC_2.0)[1]	popen(GLIBC_2.0)[1]	vswprintf(GLIBC_2.0)[1]
fgetc(GLIBC_2.0)[1]	posix_memalign(GLIBC_2.0)[1]	vswscanf(GLIBC_2.0)[1]
fgetpos(GLIBC_2.0)[1]	printf(GLIBC_2.0)[1]	vsyslog[1]
fgetpos64(GLIBC_2.1)[1]	psignal(GLIBC_2.1)[1]	vwprintf(GLIBC_2.1)[1]
fgets(GLIBC_2.0)[1]	ptsname(GLIBC_2.0)[1]	vwscanf(GLIBC_2.0)[1]

fgetwe(GLIBC_2.2)[1]	putc(GLIBC_2.2)[1]	wait(GLIBC_2.2)[1]
fgetwe_unlocked(GLIBC_2.2)[1]	putc_unlocked(GLIBC_2.2)[1]	wait3(GLIBC_2.2)[1]
fgetws(GLIBC_2.2)[1]	putchar(GLIBC_2.2)[1]	wait4(GLIBC_2.2)[1]
fileno(GLIBC_2.0)[1]	putchar_unlocked(GLIBC_2.0)[1]	waitpid(GLIBC_2.0)[1]
flock(GLIBC_2.0)[1]	putenv(GLIBC_2.0)[1]	warn(GLIBC_2.0)[1]
flockfile(GLIBC_2.0)[1]	puts(GLIBC_2.0)[1]	warnx(GLIBC_2.0)[1]
fmsg(GLIBC_2.1)[1]	pututxline(GLIBC_2.1)[1]	wepepy(GLIBC_2.1)[1]
fmatch(GLIBC_2.2.3)[1]	putw(GLIBC_2.2.3)[1]	wepnepy(GLIBC_2.2.3)[1]
fopen(GLIBC_2.1)[1]	putwe(GLIBC_2.1)[1]	wertomb(GLIBC_2.1)[1]
fopen64(GLIBC_2.1)[1]	putwchar(GLIBC_2.1)[1]	wescaseemp(GLIBC_2.1)[1]
fork(GLIBC_2.0)[1]	qsort(GLIBC_2.0)[1]	wescat(GLIBC_2.0)[1]
fpathconf(GLIBC_2.0)[1]	raise(GLIBC_2.0)[1]	weschr(GLIBC_2.0)[1]
fprintf(GLIBC_2.0)[1]	rand(GLIBC_2.0)[1]	wesemp(GLIBC_2.0)[1]
fputc(GLIBC_2.0)[1]	rand_r(GLIBC_2.0)[1]	wesecoll(GLIBC_2.0)[1]
fputs(GLIBC_2.0)[1]	random(GLIBC_2.0)[1]	wesepy(GLIBC_2.0)[1]
fputwe(GLIBC_2.2)[1]	random_r(GLIBC_2.2)[1]	wesespn(GLIBC_2.2)[1]
fputws(GLIBC_2.2)[1]	read(GLIBC_2.2)[1]	wesdup(GLIBC_2.2)[1]
fread(GLIBC_2.0)[1]	readdir(GLIBC_2.0)[1]	wesftime(GLIBC_2.0)[1]
free(GLIBC_2.0)[1]	readdir64(GLIBC_2.0)[1]	weslen(GLIBC_2.0)[1]
freeaddrinfo[1]	readdir_r[1]	wesneaseemp()[1]
freopen(GLIBC_2.0)[1]	readlink(GLIBC_2.0)[1]	wesneat(GLIBC_2.0)[1]
freopen64(GLIBC_2.1)[1]	readv(GLIBC_2.1)[1]	wesnemp(GLIBC_2.1)[1]
fscanf(GLIBC_2.0)[1]	realloc(GLIBC_2.0)[1]	wesnepy(GLIBC_2.0)[1]
fseek(GLIBC_2.0)[1]	realpath(GLIBC_2.0)[1]	wesnlen(GLIBC_2.0)[1]
fseeko(GLIBC_2.1)[1]	recv(GLIBC_2.1)[1]	wesnrtombs(GLIBC_2.1)[1]
fseeko64(GLIBC_2.1)[1]	recvfrom(GLIBC_2.1)[1]	wespbkr(GLIBC_2.1)[1]
fsetpos(GLIBC_2.0)[1]	recvmsg(GLIBC_2.0)[1]	wesrehr(GLIBC_2.0)[1]
fsetpos64(GLIBC_2.1)[1]	regcomp(GLIBC_2.1)[1]	wesrtombs(GLIBC_2.1)[1]
fstatvfs(GLIBC_2.1)[1]	regerror(GLIBC_2.1)[1]	wesspn(GLIBC_2.1)[1]
fstatvfs64(GLIBC_2.1)[1]	regexec(GLIBC_2.1)[1]	wesstr(GLIBC_2.1)[1]
fsyne(GLIBC_2.0)[1]	regfree(GLIBC_2.0)[1]	westod(GLIBC_2.0)[1]

<code>ftell(GLIBC_2.0)[1]</code>	<code>remove(GLIBC_2.0)[1]</code>	<code>westof(GLIBC_2.0)[1]</code>
<code>ftello(GLIBC_2.1)[1]</code>	<code>remque(GLIBC_2.1)[1]</code>	<code>westoimax(GLIBC_2.1)[1]</code>
<code>ftello64(GLIBC_2.1)[1]</code>	<code>rename(GLIBC_2.1)[1]</code>	<code>westok(GLIBC_2.1)[1]</code>
<code>ftime(GLIBC_2.0)[1]</code>	<code>rewind(GLIBC_2.0)[1]</code>	<code>westol(GLIBC_2.0)[1]</code>
<code>ftok(GLIBC_2.0)[1]</code>	<code>rewinddir(GLIBC_2.0)[1]</code>	<code>westold(GLIBC_2.0)[1]</code>
<code>ftruncate(GLIBC_2.0)[1]</code>	<code>rindex(GLIBC_2.0)[1]</code>	<code>westoll(GLIBC_2.0)[1]</code>
<code>ftruncate64(GLIBC_2.1)[1]</code>	<code>rmdir(GLIBC_2.1)[1]</code>	<code>westombs(GLIBC_2.1)[1]</code>
<code>ftrylockfile(GLIBC_2.0)[1]</code>	<code>sbrk(GLIBC_2.0)[1]</code>	<code>westoq(GLIBC_2.0)[1]</code>
<code>ftw(GLIBC_2.0)[1]</code>	<code>scanf(GLIBC_2.0)[1]</code>	<code>westoul(GLIBC_2.0)[1]</code>
<code>ftw64(GLIBC_2.1)[1]</code>	<code>sched_get_priority_max(GLIBC_2.1)[1]</code>	<code>westoull(GLIBC_2.1)[1]</code>
<code>funlockfile(GLIBC_2.0)[1]</code>	<code>sched_get_priority_min(GLIBC_2.0)[1]</code>	<code>westoumax(GLIBC_2.0)[1]</code>
<code>fwide(GLIBC_2.2)[1]</code>	<code>sched_getparam(GLIBC_2.2)[1]</code>	<code>westouq(GLIBC_2.2)[1]</code>
<code>fwprintf(GLIBC_2.2)[1]</code>	<code>sched_getscheduler(GLIBC_2.2)[1]</code>	<code>weswes(GLIBC_2.2)[1]</code>
<code>fwrite(GLIBC_2.0)[1]</code>	<code>sched_rr_get_interval(GLIBC_2.0)[1]</code>	<code>weswidth(GLIBC_2.0)[1]</code>
<code>fwscanf(GLIBC_2.2)[1]</code>	<code>sched_setparam(GLIBC_2.2)[1]</code>	<code>wesxfrm(GLIBC_2.2)[1]</code>
<code>gai_strerror[1]</code>	<code>sched_setscheduler()[1]</code>	<code>wetob()[1]</code>
<code>gevt(GLIBC_2.0)[1]</code>	<code>sched_yield(GLIBC_2.0)[1]</code>	<code>wetomb(GLIBC_2.0)[1]</code>
<code>getaddrinfo[1]</code>	<code>seed48()[1]</code>	<code>wetrans()[1]</code>
<code>getc(GLIBC_2.0)[1]</code>	<code>seekdir(GLIBC_2.0)[1]</code>	<code>wetype(GLIBC_2.0)[1]</code>
<code>getc_unlocked(GLIBC_2.0)[1]</code>	<code>select(GLIBC_2.0)[1]</code>	<code>wewidth(GLIBC_2.0)[1]</code>
<code>getchar(GLIBC_2.0)[1]</code>	<code>semctl(GLIBC_2.0)[1]</code>	<code>wmemchr(GLIBC_2.0)[1]</code>
<code>getchar_unlocked(GLIBC_2.0)[1]</code>	<code>semget(GLIBC_2.0)[1]</code>	<code>wmememp(GLIBC_2.0)[1]</code>
<code>getcontext(GLIBC_2.1)[1]</code>	<code>semop(GLIBC_2.1)[1]</code>	<code>wmemepy(GLIBC_2.1)[1]</code>
<code>getcwd(GLIBC_2.0)[1]</code>	<code>send(GLIBC_2.0)[1]</code>	<code>wmemmove(GLIBC_2.0)[1]</code>
<code>getdate(GLIBC_2.1)[1]</code>	<code>sendmsg(GLIBC_2.1)[1]</code>	<code>wmemset(GLIBC_2.1)[1]</code>
<code>getdomainname(GLIBC_2.0)[1]</code>	<code>sendto(GLIBC_2.0)[1]</code>	<code>wordexp(GLIBC_2.0)[1]</code>
<code>getegid(GLIBC_2.0)[1]</code>	<code>setbuf(GLIBC_2.0)[1]</code>	<code>wordfree(GLIBC_2.0)[1]</code>
<code>getenv(GLIBC_2.0)[1]</code>	<code>setbuffer(GLIBC_2.0)[1]</code>	<code>wprintf(GLIBC_2.0)[1]</code>
<code>geteuid(GLIBC_2.0)[1]</code>	<code>setcontext(GLIBC_2.0)[1]</code>	<code>write(GLIBC_2.0)[1]</code>

getgid(GLIBC_2.0)[1]	setdomainname[1]	writew(GLIBC_2.0)[1]
getgrent(GLIBC_2.0)[1]	setegid(GLIBC_2.0)[1]	wscanf(GLIBC_2.0)[1]
getgrgid(GLIBC_2.0)[1]	setenv[1]	xdr_accepted_reply(GLIBC_2.0)[1]
getgrgid_r(GLIBC_2.0)[1]	seteuid(GLIBC_2.0)[1]	xdr_array(GLIBC_2.0)[1]
getgrnam(GLIBC_2.0)[1]	setgid(GLIBC_2.0)[1]	xdr_bool(GLIBC_2.0)[1]
getgrnam_r(GLIBC_2.0)[1]	setgrent(GLIBC_2.0)[1]	xdr_bytes(GLIBC_2.0)[1]
getgroups(GLIBC_2.0)[1]	setgroups(GLIBC_2.0)[1]	xdr_callhdr(GLIBC_2.0)[1]
gethostbyaddr(GLIBC_2.0)[1]	sethostid(GLIBC_2.0)[1]	xdr_callmsg(GLIBC_2.0)[1]
gethostbyname(GLIBC_2.0)[1]	sethostname(GLIBC_2.0)[1]	xdr_char(GLIBC_2.0)[1]
gethostid(GLIBC_2.0)[1]	setitimer(GLIBC_2.0)[1]	xdr_double(GLIBC_2.0)[1]
gethostname(GLIBC_2.0)[1]	setlocale(GLIBC_2.0)[1]	xdr_enum(GLIBC_2.0)[1]
getitimer(GLIBC_2.0)[1]	setlogmask(GLIBC_2.0)[1]	xdr_float(GLIBC_2.0)[1]
getloadavg(GLIBC_2.2)[1]	setnetent(GLIBC_2.2)[1]	xdr_free(GLIBC_2.2)[1]
getlogin(GLIBC_2.0)[1]	setpgid(GLIBC_2.0)[1]	xdr_int(GLIBC_2.0)[1]
getnameinfo[1]	setpgid[1]	xdr_long[1]
getnetbyaddr(GLIBC_2.0)[1]	setpriority(GLIBC_2.0)[1]	xdr_opaque(GLIBC_2.0)[1]
getopt(GLIBC_2.0)[1]	setprotoent(GLIBC_2.0)[1]	xdr_opaque_auth(GLIBC_2.0)[1]
getopt_long(GLIBC_2.0)[1]	setpwent(GLIBC_2.0)[1]	xdr_pointer(GLIBC_2.0)[1]
getopt_long_only(GLIBC_2.0)[1]	setregid(GLIBC_2.0)[1]	xdr_reference(GLIBC_2.0)[1]
getpagesize(GLIBC_2.0)[1]	setreuid(GLIBC_2.0)[1]	xdr_rejected_reply(GLIBC_2.0)[1]
getpeername(GLIBC_2.0)[1]	setrlimit(GLIBC_2.0)[1]	xdr_replymsg(GLIBC_2.0)[1]
getpgid(GLIBC_2.0)[1]	setrlimit64[1]	xdr_short(GLIBC_2.0)[1]
getpgrp(GLIBC_2.0)[1]	setservent(GLIBC_2.0)[1]	xdr_string(GLIBC_2.0)[1]
getpid(GLIBC_2.0)[1]	setsid(GLIBC_2.0)[1]	xdr_u_char(GLIBC_2.0)[1]
getppid(GLIBC_2.0)[1]	setsockopt(GLIBC_2.0)[1]	xdr_u_int(GLIBC_2.0)[1]
getpriority(GLIBC_2.0)[1]	setstate(GLIBC_2.0)[1]	xdr_u_long(GLIBC_2.0)[1]
getprotobyname(GLIBC_2.0)[1]	setuid(GLIBC_2.0)[1]	xdr_u_short(GLIBC_2.0)[1]
getprotobynumber(GLIBC_2.0)[1]	setutent(GLIBC_2.0)[1]	xdr_union(GLIBC_2.0)[1]
getprotoent(GLIBC_2.0)[1]	setutxent(GLIBC_2.0)[1]	xdr_vector(GLIBC_2.0)[1]
getpwent(GLIBC_2.0)[1]	setvbuf(GLIBC_2.0)[1]	xdr_void(GLIBC_2.0)[1]
getpwnam(GLIBC_2.0)[1]	shmat(GLIBC_2.0)[1]	xdr_wrapstring(GLIBC_2.0)[1]

getpwnam_r(GLIBC_2.0)[1]	shmctl(GLIBC_2.0)[1]	xdrmem_create(GLIBC_2.0)[1]
getpwuid(GLIBC_2.0)[1]	shmdt(GLIBC_2.0)[1]	xdrrec_create(GLIBC_2.0)[1]
getpwuid_r(GLIBC_2.0)[1]	shmget(GLIBC_2.0)[1]	xdrrec_eof(GLIBC_2.0)[1]
getrlimit(GLIBC_2.2)[1]	shutdown(GLIBC_2.2)[1]	
getrlimit64(GLIBC_2.1)[1]	sigaction(GLIBC_2.1)[1]	

Table A-18. libc Data Interfaces

__daylight	__timezone	_sys_errlist
__environ	__tzname	

A.14. libpthread

The behaviour of the interfaces in this library is specified by the following Standards.

Large File Support

Linux Standard Base

ISO/IEC 9945:2003 Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3

Table A-19. libpthread Function Interfaces

_pthread_cleanup_pop[1]	pthread_create()[1]	pthread_rwlock_trywrlock()[1]
_pthread_cleanup_push[1]	pthread_detach()[1]	pthread_rwlock_unlock()[1]
pread(GLIBC_2.1)[1]	pthread_equal(GLIBC_2.1)[1]	pthread_rwlock_wrlock(GLIBC_2.1)[1]
pread64(GLIBC_2.1)[1]	pthread_exit(GLIBC_2.1)[1]	pthread_rwlockattr_destroy(GLIBC_2.1)[1]
pthread_attr_destroy(GLIBC_2.0)[1]	pthread_getspecific(GLIBC_2.0)[1]	pthread_rwlockattr_getpshared(GLIBC_2.0)[1]
pthread_attr_getdetachstate(GLIBC_2.0)[1]	pthread_join(GLIBC_2.0)[1]	pthread_rwlockattr_init(GLIBC_2.0)[1]
pthread_attr_getguardsize(GLIBC_2.1)[1]	pthread_key_create(GLIBC_2.1)[1]	pthread_rwlockattr_setpshared(GLIBC_2.1)[1]
pthread_attr_getschedparam(GLIBC_2.0)[1]	pthread_key_delete(GLIBC_2.0)[1]	pthread_self(GLIBC_2.0)[1]
pthread_attr_getstackaddr(GLIBC_2.1)[1]	pthread_kill(GLIBC_2.1)[1]	pthread_setcancelstate(GLIBC_2.1)[1]
pthread_attr_getstacksize(GLIBC_2.1)[1]	pthread_mutex_destroy(GLIBC_2.1)[1]	pthread_setcanceltype(GLIBC_2.1)[1]

<code>pthread_attr_init(GLIBC_2.1)[1]</code>	<code>pthread_mutex_init(GLIBC_2.1)[1]</code>	<code>pthread_setconcurrency[1]</code>
<code>pthread_attr_setdetachstate(GLIBC_2.0)[1]</code>	<code>pthread_mutex_lock(GLIBC_2.0)[1]</code>	<code>pthread_setspecific(GLIBC_2.0)[1]</code>
<code>pthread_attr_setguardsize(GLIBC_2.1)[1]</code>	<code>pthread_mutex_trylock(GLIBC_2.1)[1]</code>	<code>pthread_sigmask(GLIBC_2.1)[1]</code>
<code>pthread_attr_setschedparam(GLIBC_2.0)[1]</code>	<code>pthread_mutex_unlock(GLIBC_2.0)[1]</code>	<code>pthread_testcancel(GLIBC_2.0)[1]</code>
<code>pthread_attr_setstackaddr(GLIBC_2.1)[1]</code>	<code>pthread_mutexattr_destroy(GLIBC_2.1)[1]</code>	<code>pwrite(GLIBC_2.1)[1]</code>
<code>pthread_attr_setstacksize(GLIBC_2.1)[1]</code>	<code>pthread_mutexattr_getpshared(GLIBC_2.1)[1]</code>	<code>pwrite64(GLIBC_2.1)[1]</code>
<code>pthread_cancel(GLIBC_2.0)[1]</code>	<code>pthread_mutexattr_gettype(GLIBC_2.0)[1]</code>	<code>sem_close(GLIBC_2.0)[1]</code>
<code>pthread_cond_broadcast(GLIBC_2.0)[1]</code>	<code>pthread_mutexattr_init(GLIBC_2.0)[1]</code>	<code>sem_destroy(GLIBC_2.0)[1]</code>
<code>pthread_cond_destroy(GLIBC_2.0)[1]</code>	<code>pthread_mutexattr_setpshared(GLIBC_2.0)[1]</code>	<code>sem_getvalue(GLIBC_2.0)[1]</code>
<code>pthread_cond_init(GLIBC_2.0)[1]</code>	<code>pthread_mutexattr_settype(GLIBC_2.0)[1]</code>	<code>sem_init(GLIBC_2.0)[1]</code>
<code>pthread_cond_signal(GLIBC_2.0)[1]</code>	<code>pthread_once(GLIBC_2.0)[1]</code>	<code>sem_open(GLIBC_2.0)[1]</code>
<code>pthread_cond_timedwait(GLIBC_2.0)[1]</code>	<code>pthread_rwlock_destroy(GLIBC_2.0)[1]</code>	<code>sem_post(GLIBC_2.0)[1]</code>
<code>pthread_cond_wait(GLIBC_2.0)[1]</code>	<code>pthread_rwlock_init(GLIBC_2.0)[1]</code>	<code>sem_timedwait(GLIBC_2.0)[1]</code>
<code>pthread_condattr_destroy(GLIBC_2.0)[1]</code>	<code>pthread_rwlock_rdlock(GLIBC_2.0)[1]</code>	<code>sem_trywait(GLIBC_2.0)[1]</code>
<code>pthread_condattr_getpshared[1]</code>	<code>pthread_rwlock_timedrdlock[1]</code>	<code>sem_unlink()[1]</code>
<code>pthread_condattr_init(GLIBC_2.0)[1]</code>	<code>pthread_rwlock_timedwrlock[1]</code>	<code>sem_wait(GLIBC_2.0)[1]</code>
<code>pthread_condattr_setpshared[1]</code>	<code>pthread_rwlock_tryrdlock()[1]</code>	

A.15. libpam

The behaviour of the interfaces in this library is specified by the following Standards.

Linux Standard Base

Table A-20. libpam Function Interfaces

pam_acct_mgmt[1]	pam_fail_delay[1]	pam_setcred[1]
pam_authenticate[1]	pam_get_item[1]	pam_start[1]
pam_chautok[1]	pam_getenvlist[1]	pam_strerror[1]
pam_close_session[1]	pam_open_session[1]	
pam_end[1]	pam_set_item[1]	

Linux Packaging Specification

2

3 **Linux Packaging Specification**

Table of Contents

I. Package Format and Installation	405
1. Software Installation	1
1.1. Package File Format	1
1.1.1. Lead Section	1
1.1.2. Header Structure.....	2
1.1.2.1. Header Record	2
1.1.2.2. Index Record.....	3
1.1.2.2.1. Index Type Values	3
1.1.2.2.2. Index Tag Values	4
1.1.2.3. Header Store	5
1.1.3. Signature Section	5
1.1.4. Header Section	7
1.1.4.1. Package Information	7
1.1.4.2. Installation Information	9
1.1.4.3. File Information	10
1.1.4.4. Dependency Information	12
1.1.4.4.1. Package Dependency Values.....	14
1.1.4.4.2. Package Dependencies Attributes	15
1.1.4.5. Other Information	15
1.1.5. Payload Section	17
1.2. Package Script Restrictions	19
1.3. Package Tools.....	19
1.4. Package Naming	19
1.5. Package Dependencies.....	20
1.6. Package Architecture Considerations	20

List of Tables

1-1. RPM File Format	1
1-2. Signature Format	2
1-3. Index Type values.....	3
1-4. Header Private Tag Values	4
1-5. Signature Tag Values	5
1-6. Signature Digest Tag Values	6
1-7. Signature Signing Tag Values	6
1-8. Package Info Tag Values	7
1-9. Installation Tag Values.....	9
1-10. File Info Tag Values	10
1-11. Package Dependency Tag Values.....	12
1-12. Index Type values.....	14
1-13. Package Dependency Attributes	15
1-14. Other Tag Values.....	15
1-15. CPIO File Format	17

I. Package Format and Installation

Chapter 1. Software Installation

Applications shall either be packaged in the RPM packaging format as defined in this specification, or supply an installer which is LSB conforming (for example, calls LSB commands and utilities).¹

Distributions shall provide a mechanism for installing applications in this packaging format with some restrictions listed below.²

1.1. Package File Format

An RPM format file consists of 4 sections, the Lead, Signature, Header, and the Payload. All values are stored in network byte order.

Table 1-1. RPM File Format

Lead
Signature
Header
Payload

These 4 sections shall exist in the order specified.

The lead section is used to identify the package file.

The signature section is used to verify the integrity, and optionally, the authenticity of the majority of the package file.

The header section contains all available information about the package. Entries such as the package's name, version, and file list, are contained in the header.

The payload section holds the files to be install.

1.1.1. Lead Section

```
struct rpmlead {
    unsigned char magic[4];
    unsigned char major, minor;
    short type;
    short archnum;
    char name[66];
    short osnum;
    short signature_type;
    char reserved[16];
} ;
```

magic

Value identifying this file as an RPM format file. This value shall be "\355\253\356\333".

27 *major*
 28 Value indicating the major version number of the file format version. This value shall be 3.

29 *minor*
 30 Value indicating the minor revision number of file format version. This value shall be 0.

31 *type*
 32 Value indicating whether this is a source or binary package. This value shall be 0 to indicate a binary package.

33 *archnum*
 34 Value indicating the architecture for which this package is valid. This value is specified in the
 35 architecture-specific LSB specification.

36 *name*
 37 A NUL terminated string that provides the package name. This name shall conform with the Package Naming
 38 section of this specification.

39 *osnum*
 40 Value indicating the Operating System for which this package is valid. This value shall be 1.

41 *signature_type*
 42 Value indicating the type of the signature used in the Signature part of the file. This value shall be 5.

43 *reserved*
 44 Reserved space. The value is undefined.

1.1.2. Header Structure

45 The Header structure is used for both the Signature and Header Sections. A Header Structure consists of 3 parts, a
 46 Header record, followed by 1 or more Index records, followed by 0 or more bytes of data associated with the Index
 47 records. A Header structure shall be aligned to an 8 byte boundary.

48 **Table 1-2. Signature Format**

Header Record
Array of Index Records
Store of Index Values

1.1.2.1. Header Record

```

51 struct rpmheader {
52     unsigned char magic[4];
53     unsigned char reserved[4];
54     int nindex;
55     int hsize;
56     } ;
  
```


57 *magic*

58 Value identifying this record as an RPM header record. This value shall be "\216\255\350\001".

59 *reserved*

60 Reserved space. This value shall be "\000\000\000\000".

61 *nindex*

62 The number of Index Records that follow this Header Record. There should be at least 1 Index Record.

63 *hsize*

64 The size in bytes of the storage area for the data pointed to by the Index Records.

65 1.1.2.2. Index Record

```
66 struct rpmhdrindex {
67     int tag;
68     int type;
69     int offset;
70     int count;
71 } ;
```

72 *tag*

73 Value identifying the purpose of the data associated with this Index Record. This value of this field is dependent
74 on the context in which the Index Record is used, and is defined below and in later sections.

75 *type*

76 Value identifying the type of the data associated with this Index Record. The possible *type* values are defined
77 below.

78 *offset*

79 Location in the Store of the data associated with this Index Record. This value should be between 0 and the value
80 contained in the *hsize* of the Header Structure.

81 *count*

82 Size of the data associated with this Index Record. The *count* is the number of elements whose size is defined
83 by the type of this Record.

84 1.1.2.2.1. Index Type Values

85 The possible values for the *type* field are defined in this table.

86 **Table 1-3. Index Type values**

Type	Value	Size (in bytes)	Alignment
RPM_NULL_TYPE	0	Not Implemented.	
RPM_CHAR_TYPE	1	1	1

Type	Value	Size (in bytes)	Alignment
RPM_INT8_TYPE	2	1	1
RPM_INT16_TYPE	3	2	2
RPM_INT32_TYPE	4	4	4
RPM_INT64_TYPE	5	Reserved.	
RPM_STRING_TYPE	6	variable, NUL terminated	1
RPM_BIN_TYPE	7	1	1
RPM_STRING_ARRAY_TYPE	8	Variable, sequence of NUL terminated strings	1
RPM_I18NSTRING_TYPE	9	variable, sequence of NUL terminated strings	1

The string arrays specified for entries of type `RPM_STRING_ARRAY_TYPE` and `RPM_I18NSTRING_TYPE` are vectors of strings in a contiguous block of memory, each element separated from its neighbors by a NUL character.

Index records with type `RPM_I18NSTRING_TYPE` shall always have a *count* of 1. The array entries in an index of type `RPM_I18NSTRING_TYPE` correspond to the locale names contained in the `RPMTAG_HDRI18N` index.

1.1.2.2.2. Index Tag Values

Some values are designated as header private, and may appear in any header structure. These are defined here. Additional values are defined in later sections.

Table 1-4. Header Private Tag Values

Name	Tag Value	Type	Count	Status
RPMTAG_HEADERSIGNATURES	62	BIN	16	Optional
RPMTAG_HEADERIMMUTABLE	63	BIN	16	Optional
RPMTAG_HEADERI18N	100	STRING_ARRAY		Required

RPMTAG_HEADERSIGNATURES

The signature tag differentiates a signature header from a metadata header, and identifies the original contents of the signature header.

RPMTAG_HEADERIMMUTABLE

This tag contains an index record which specifies the portion of the Header Record which was used for the calculation of a signature. This data shall be preserved or any header-only signature will be invalidated.

RPMTAG_HEADER18NTABLE

Contains a list of locales for which strings are provided in other parts of the package.

Not all Index records defined here will be present in all packages. Each tag value has a status which is defined here.

Required

This Index Record shall be present.

Optional

This Index Record may be present.

Deprecated

This Index Record should not be present.

Obsolete

This Index Record shall not be present.

Reserved

This Index Record shall not be present.

1.1.2.3. Header Store

The header store contains the values specified by the Index structures. These values are aligned according to their type and padding is used if needed. The store is located immediately following the Index structures.

1.1.3. Signature Section

The Signature section is implemented using the Header structure. The signature section defines the following additional tag values which may be used in the Index structures.

These values exist to provide additional information about the rest of the package.

Table 1-5. Signature Tag Values

Name	Tag Value	Type	Count	Status
SIGTAG_SIGSIZE	1000	INT32	1	Required
SIGTAG_PAYLOADSIZE	1007	INT32	1	Optional

SIGTAG_SIGSIZE

This tag specifies the combined size of the Header and Payload sections.

SIGTAG_PAYLOADSIZE

This tag specifies the uncompressed size of the Payload archive, including the cpio headers.

These values exist to ensure the integrity of the rest of the package.

Table 1-6. Signature Digest Tag Values

Name	Tag Value	Type	Count	Status
SIGTAG_MD5	1004	BIN	16	Required
SIGTAG_SHA1HEADER	1010	STRING	1	Optional

SIGTAG_MD5

This tag specifies the 128-bit MD5 checksum of the combined Header and Archive sections.

SIGTAG_SHA1HEADER

This index contains the SHA1 checksum of the entire Header Section, including the Header Record, Index Records and Header store.

These values exist to provide authentication of the package.

Table 1-7. Signature Signing Tag Values

Name	Tag Value	Type	Count	Status
SIGTAG_PGP	1002	BIN	1	Optional
SIGTAG_GPG	1005	BIN	65	Optional
SIGTAG_DSAHEADER	1011	BIN	1	Optional
SIGTAG_RSAHEADER	1012	BIN	1	Optional

SIGTAG_PGP

This tag specifies the RSA signature of the combined Header and Payload sections. The data is formatted as a Version 3 Signature Packet as specified in RFC 2440: OpenPGP Message Format.

SIGTAG_GPG

The tag contains the DSA signature of the combined Header and Payload sections. The data is formatted as a Version 3 Signature Packet as specified in RFC 2440: OpenPGP Message Format.

SIGTAG_DSAHEADER

The tag contains the DSA signature of the Header section. The data is formatted as a Version 3 Signature Packet as specified in RFC 2440: OpenPGP Message Format. If this tag is present, then the SIGTAG_GPG tag shall also be present.

SIGTAG_RSAHEADER

The tag contains the RSA signature of the Header section. The data is formatted as a Version 3 Signature Packet as specified in RFC 2440: OpenPGP Message Format. If this tag is present, then the SIGTAG_PGP shall also be present.

1.1.4. Header Section

The Header section is implemented using the Header structure. The Header section defines the following additional tag values which may be used in the Index structures.

1.1.4.1. Package Information

The following tag values are used to indicate information that describes the package as a whole.

Table 1-8. Package Info Tag Values

Name	Tag Value	Type	Count	Status
RPMTAG_NAME	1000	STRING	1	Required
RPMTAG_VERSION	1001	STRING	1	Required
RPMTAG_RELEASE	1002	STRING	1	Required
RPMTAG_SUMMARY	1004	I18NSTRING	1	Required
RPMTAG_DESCRIPTION	1005	I18NSTRING	1	Required
RPMTAG_SIZE	1009	INT32	1	Required
RPMTAG_LICENSE	1014	STRING	1	Required
RPMTAG_GROUP	1016	I18NSTRING	1	Required
RPMTAG_OS	1021	STRING	1	Required
RPMTAG_ARCH	1022	STRING	1	Required
RPMTAG_SOURCE RPM	1044	STRING	1	Optional
RPMTAG_ARCHIVESIZE	1046	INT32	1	Optional
RPMTAG_RPMVERSION	1064	STRING	1	Optional
RPMTAG_COOKIE	1094	STRING	1	Optional
RPMTAG_PAYLOADFORMAT	1124	STRING	1	Required
RPMTAG_PAYLOADCOMPRESSOR	1125	STRING	1	Required

Name	Tag Value	Type	Count	Status
RPMTAG_PAYLOAD ADFLAGS	1126	STRING	1	Required

158

159 RPMTAG_NAME

160 This tag specifies the name of the package.

161 RPMTAG_VERSION

162 This tag specifies the version of the package.

163 RPMTAG_RELEASE

164 This tag specifies the release of the package.

165 RPMTAG_SUMMARY

166 This tag specifies the summary description of the package. The summary value pointed to by this index record
 167 contains a one line description of the package.

168 RPMTAG_DESCRIPTION

169 This tag specifies the description of the package. The description value pointed to by this index record contains a
 170 full description of the package.

171 RPMTAG_SIZE

172 This tag specifies the sum of the sizes of the regular files in the archive.

173 RPMTAG_LICENSE

174 This tag specifies the license which applies to this package.

175 RPMTAG_GROUP

176 This tag specifies the administrative group to which this package belongs.

177 RPMTAG_OS

178 This tag specifies the OS of the package. The OS value pointed to by this index record shall be "linux".

179 RPMTAG_ARCH

180 This tag specifies the architecture of the package. The architecture value pointed to by this index record is defined
 181 in architecture specific LSB specification.

182 RPMTAG_SOURCERPM

183 This tag specifies the name of the source RPM

184 RPMTAG_ARCHIVE_SIZE

185 This tag specifies the uncompressed size of the Payload archive, including the cpio headers.

RPMTAG_RPMVERSION

This tag indicates the version of RPM tool used to build this package. The value is unused.

RPMTAG_COOKIE

This tag contains an opaque string whose contents are undefined.

RPMTAG_PAYLOADFORMAT

This tag specifies the format of the Archive section. The format value pointed to by this index record shall be 'cpio'.

RPMTAG_PAYLOADCOMPRESSOR

This tag specifies the compression used on the Archive section. The compression value pointed to by this index record shall be 'gzip'

RPMTAG_PAYLOADFLAGS

This tag indicates the compression level used for the Payload. This value shall always be '9'.

1.1.4.2. Installation Information

The following tag values are used to provide information needed during the installation of the package.

Table 1-9. Installation Tag Values

Name	Tag Value	Type	Count	Status
RPMTAG_PREIN	1023	STRING	1	Optional
RPMTAG_POSTIN	1024	STRING	1	Optional
RPMTAG_PREUN	1025	STRING	1	Optional
RPMTAG_POSTUN	1026	STRING	1	Optional
RPMTAG_PREINPROG	1085	STRING	1	Optional
RPMTAG_POSTINPROG	1086	STRING	1	Optional
RPMTAG_PREUNPROG	1087	STRING	1	Optional
RPMTAG_POSTUNPROG	1088	STRING	1	Optional

RPMTAG_PREIN

This tag specifies the preinstall scriptlet.

RPMTAG_POSTIN

This tag specifies the postinstall scriptlet.

206 **RPMTAG_PREUN**
 207 his tag specifies the preuninstall scriptlet.

208 **RPMTAG_POSTUN**
 209 This tag specified the postuninstall scriptlet.

210 **RPMTAG_PREINPROG**
 211 This tag specifies the name of the interpreter to which the preinstall scriptlet will be passed. The interpreter pointed
 212 to by this index record shall be '/bin/sh'.

213 **RPMTAG_POSTINPROG**
 214 This tag specifies the name of the interpreter to which the postinstall scriptlet will be passed. The interpreter
 215 pointed to by this index record shall be '/bin/sh'.

216 **RPMTAG_PREUNPROG**
 217 This tag specifies the name of the interpreter to which the preuninstall scriptlet will be passed. The interpreter
 218 pointed to by this index record shall be '/bin/sh'.

219 **RPMTAG_POSTUNPROG**
 220 This program specifies the name of the interpreter to which the postuninstall scriptlet will be passed. The
 221 interpreter pointed to by this index record shall be '/bin/sh'.

222 1.1.4.3. File Information

223 The following tag values are used to provide information about the files in the payload. This information is provided in
 224 the header to allow more efficient access of the information.

225 **Table 1-10. File Info Tag Values**

Name	Tag Value	Type	Count	Status
RPMTAG_OLDFILENAMES	1027	STRING_ARRAY		Optional
RPMTAG_FILESIZE	1028	INT32		Required
RPMTAG_FILEMODES	1030	INT16		Required
RPMTAG_FILERDV	1033	INT16		Required
RPMTAG_FILEMTIMES	1034	INT32		Required
RPMTAG_FILEMD5	1035	STRING_ARRAY		Required
RPMTAG_FILEL	1036	STRING_ARRAY		Required

Name	Tag Value	Type	Count	Status
NKTOS				
RPMTAG_FILEFL AGS	1037	INT32		Required
RPMTAG_FILEUS ERNAME	1039	STRING_ARRAY		Required
RPMTAG_FILEGR OUPNAME	1040	STRING_ARRAY		Required
RPMTAG_FILEDE VICES	1095	INT32		Required
RPMTAG_FILEIN ODES	1096	INT32		Required
RPMTAG_FILELA NGS	1097	STRING_ARRAY		Required
RPMTAG_DIRIND EXES	1116	INT32		Optional
RPMTAG_BASEN AMES	1117	STRING_ARRAY		Optional
RPMTAG_DIRNA MES	1118	STRING_ARRAY		Optional

226

227 RPMTAG_OLDFILENAMES

228 This tag specifies the filenames when not in a compressed format as determined by the absence of
 229 rpmlib(CompressedFileNames) in the RPMTAG_REQUIRENAME index.

230 RPMTAG_FILESIZES

231 This tag specifies the size of each file in the archive.

232 RPMTAG_FILEMODES

233 This tag specifies the mode of each file in the archive.

234 RPMTAG_FILERDEVS

235 This tag specifies the device number from which the file was copied.

236 RPMTAG_FILEMTIMES

237 This tag specifies the modification time in seconds since the epoch of each file in the archive.

238 RPMTAG_FILEMD5S

239 This tag specifies the ASCII representation of the MD5 sum of the corresponding file contents. This value is
 240 empty if the corresponding archive entry is not a regular file.

241 **RPMTAG_FILELINKTOS**
 242 The target for a symlink, otherwise NULL.

243 **RPMTAG_FILEFLAGS**
 244 This tag specifies the bit(s) to classify and control how files are to be installed.

245 **RPMTAG_FILEUSERNAME**
 246 This tag specifies the owner of the corresponding file.

247 **RPMTAG_FILEGROUPNAME**
 248 This tag specifies the of the corresponding file.

249 **RPMTAG_FILEDEVICES**
 250 This tag specifies the 16 bit device number from which the file was copied.

251 **RPMTAG_FILEINODES**
 252 This tag specifies the inode value from the original file on the build host.

253 **RPMTAG_FILELANGS**
 254 This tag specifies a per-file locale marker used to install only locale specific subsets of files when the package is
 255 installed.

256 **RPMTAG_DIRINDEXES**
 257 This tag specifies the index into the array provided by the **RPMTAG_DIRNAMES** Index which contains the
 258 directory name for the corresponding filename.

259 **RPMTAG_BASENAMES**
 260 This tag specifies the base portion of the corresponding filename.

261 **RPMTAG_DIRNAMES**
 262 This tag specifies the directory portion of the corresponding filename. Each directory name shall contain a
 263 trailing '/'.
 264 One of **RPMTAG_OLDFILENAMES** or the tuple
 265 **RPMTAG_DIRINDEXES,RPMTAG_BASENAMES,RPMTAG_DIRNAMES** shall be present, but not both.

266 **1.1.4.4. Dependency Information**

267 The following tag values are used to provide information about interdependencies between packages.

268 **Table 1-11. Package Dependency Tag Values**

Name	Tag Value	Type	Count	Status
RPMTAG_PROVIDENAME	1047	STRING_ARRAY	1	Required
RPMTAG_REQUIRENAME	1048	INT32		Required

Name	Tag Value	Type	Count	Status
REFLAGS				
RPMTAG_REQUIRENAME	1049	STRING_ARRAY		Required
RPMTAG_REQUIREVERSION	1050	STRING_ARRAY		Required
RPMTAG_CONFLICTFLAGS	1053	INT32		Optional
RPMTAG_CONFLICTNAME	1054	STRING_ARRAY		Optional
RPMTAG_CONFLICTVERSION	1055	STRING_ARRAY		Optional
RPMTAG_OBSOLETENAME	1090	STRING_ARRAY		Optional
RPMTAG_PROVIDEFLAGS	1112	INT32		Required
RPMTAG_PROVIDEVERSION	1113	STRING_ARRAY		Required
RPMTAG_OBSOLETEFLAGS	1114	INT32	1	Optional
RPMTAG_OBSOLETEVERSION	1115	STRING_ARRAY		Optional

269

270 RPMTAG_PROVIDENAME

271 This tag indicates the name of the dependency provided by this package.

272 RPMTAG_REQUIREFLAGS

273 Bits(s) to specify the dependency range and context.

274 RPMTAG_REQUIRENAME

275 This tag indicates the dependencies for this package.

276 RPMTAG_REQUIREVERSION

277 This tag indicates the versions associated with the values found in the RPMTAG_REQUIRENAME Index.

278 RPMTAG_CONFLICTFLAGS

279 Bits(s) to specify the conflict range and context.

280 RPMTAG_CONFLICTNAME

281 This tag indicates the conflictind dependencies for this package.

282 RPMTAG_CONFLICTVERSION

283 This tag indicates the versions associated with the values found in the RPMTAG_CONFLICTNAME Index.

284 RPMTAG_OBSOLETE_NAME

285 This tag indicates the obsoleted dependencies for this package.

286 RPMTAG_PROVIDEFLAGS

287 Bits(s) to specify the conflict range and context.

288 RPMTAG_PROVIDEVERSION

289 This tag indicates the versions associated with the values found in the RPMTAG_PROVIDENAME Index.

290 RPMTAG_OBSOLETEFLAGS

291 Bits(s) to specify the conflict range and context.

292 RPMTAG_OBSOLETEVERSION

293 This tag indicates the versions associated with the values found in the RPMTAG_OBSOLETE_NAME Index.

294 1.1.4.4.1. Package Dependency Values

295 The package dependencies are stored in the RPMTAG_REQUIRENAME and RPMTAG_REQUIREVERSION index records.

296 The following values may be used.

297 **Table 1-12. Index Type values**

Name	Version	Meaning	Status
lsb	2.0	Indicates this is an LSB conforming package.	Required
rpmlib(VersionedDependencies)	3.0.3-1	Indicates That the package contains PMTAG_PROVIDENAME, RPMTAG_OBSOLETE_NAME or RPMTAG_PREREQ records that have a version associated with them.	Optional
rpmlib(PayloadFilesHavePrefix)	4.0-1	Indicates the filenames in the Archive have had "." prepended to them.	Optional
rpmlib(CompressedFileNames)	3.0.4-1	Indicates that the filenames in the Payload are represented in the RPMTAG_DIRINDEXES, RPMTAG_DIRNAME and	Optional

Name	Version	Meaning	Status
		RPMTAG_BASENAME S indexes.	
/bin/sh		Interpreter usually required for installation scripts.	Optional

1.1.4.4.2. Package Dependencies Attributes

The package dependency attributes are stored in the RPMTAG_REQUIREFLAGS, RPMTAG_PROVIDEFLAGS and RPMTAG_OBSOLETEFLAGS index records. The following values may be used.

Table 1-13. Package Dependency Attributes

Name	Value	Meaning
RPMSSENSE_LESS	0x02	
RPMSSENSE_GREATER	0x04	
RPMSSENSE_EQUAL	0x08	
RPMSSENSE_PREREQ	0x40	
RPMSSENSE_INTERP	0x100	
RPMSSENSE_SCRIPT_PRE	0x200	
RPMSSENSE_SCRIPT_POST	0x400	
RPMSSENSE_SCRIPT_PREUN	0x800	
RPMSSENSE_SCRIPT_POSTUN	0x1000	
RPMSSENSE_RPMLIB	0x1000000	

1.1.4.5. Other Information

The following tag values are also found in the Header section.

Table 1-14. Other Tag Values

Name	Tag Value	Type	Count	Status
RPMTAG_BUILD TIME	1006	INT32	1	Optional
RPMTAG_BUILD HOST	1007	STRING	1	Optional
RPMTAG_FILEVE RIFYFLAGS	1045	INT32		Optional
RPMTAG_CHANG	1080	INT32		Optional

Name	Tag Value	Type	Count	Status
ELOGTIME				
RPMTAG_CHANG ELOGNAME	1081	STRING_ARRAY		Optional
RPMTAG_CHANG ELOGTEXT	1082	STRING_ARRAY		Optional
RPMTAG_OPTFL AGS	1122	STRING	1	Optional
RPMTAG_RHNPL ATFORM	1131	STRING	1	Deprecated
RPMTAG_PLATF ORM	1132	STRING	1	Optional

307

308 RPMTAG_BUILDTIME

309 This tag specifies the time as seconds since the epoch at which the package was built.

310 RPMTAG_BUILDHOST

311 This tag specifies the on which which the package was built.

312 RPMTAG_FILEVERIFYFLAGS

313 This tag specifies the bit(s) to control how files are to be verified after install, specifying which checks should be
314 performed.

315 RPMTAG_CHANGELOGTIME

316 This tag specifies the Unix time in seconds since the epoch associated with each entry in the Changelog file.

317 RPMTAG_CHANGELOGNAME

318 This tag specifies the name of who made a change to this package

319 RPMTAG_CHANGELOGTEXT

320 This tag specifies the changes associated with a changelog entry.

321 RPMTAG_OPTFLAGS

322 This tag indicates additional flags which may have been passed to the compiler when building this package.

323 RPMTAG_RHNPLATFORM

324 This tag contains an opaque string whose contents are undefined.

325 RPMTAG_PLATFORM

326 This tag contains an opaque string whose contents are undefined.

1.1.5. Payload Section

The Payload section contains a compressed cpio archive. The format of this section is defined by ~~RFC 1952: GZIP file format specification version 4.3~~RFC 1952: GZIP File Format Specification.

When uncompressed, the cpio archive contains a sequence of records for each file. Each record contains a CPIO Header, Filename, Padding, and File Data.

Table 1-15. CPIO File Format

CPIO Header	Header structure as defined below.
Filename	NUL terminated ASCII string containing the name of the file.
Padding	0-3 bytes as needed to align the file stream to a 4 byte boundary.
File data	The contents of the file.
Padding	0-3 bytes as needed to align the file stream to a 4 byte boundary.

The CPIO Header uses the following header structure (sometimes referred to as "new ASCII" or "SVR4 cpio"). All numbers are stored as ASCII representations of their hexadecimal value with leading zeros as needed to fill the field. With the exception of *c_namesize* and the corresponding name string, and *c_checksum*, all information contained in the CPIO Header is also represented in the Header Section. The values in in the CPIO Header shall match the values contained in the Header Section.

```

struct {
    char    c_magic[6];
    char    c_ino[8];
    char    c_mode[8];
    char    c_uid[8];
    char    c_gid[8];
    char    c_nlink[8];
    char    c_mtime[8];
    char    c_filesize[8];
    char    c_devmajor[8];
    char    c_devminor[8];
    char    c_rdevmajor[8];
    char    c_rdevminor[8];
    char    c_namesize[8];
    char    c_checksum[8];
};

```

c_magic

Value identifying this cpio format. This value shall be "070701".

356 *c_ino*

357 This field contains the inode number from the filesystem from which the file was read. This field is ignored when
 358 installing a package. This field shall match the corresponding value in the `RPMTAG_FILEINODES` index in the
 359 Header section.

360 *c_mode*

361 Permission bits of the file. This is an ascii representation of the hexadecimal number representing the bit as
 362 defined for the *st_mode* field of the `stat` structure defined for the `stat` function. This field shall match the
 363 corresponding value in the `RPMTAG_FILEMODES` index in the Header section.

364 *c_uid*

365 Value identifying this owner of this file. This value matches the `uid` value of the corresponding user in the
 366 `RPMTAG_FILEUSERNAME` as found on the system where this package was built. The username specified in
 367 `RPMTAG_FILEUSERNAME` should take precedence when installing the package.

368 *c_gid*

369 Value identifying this group of this file. This value matches the `gid` value of the corresponding user in the
 370 `RPMTAG_FILEGROUPNAME` as found on the system where this package was built. The groupname specified
 371 in `RPMTAG_FILEGROUPNAME` should take precedence when installing the package.

372 *c_nlink*

373 Value identifying the number of links associated with this file. If the value is greater than 1, then this filename
 374 will be linked to 1 or more files in this archive that has a matching value for the `c_ino`, `c_devmajor` and
 375 `c_devminor` fields.

376 *c_mtime*

377 Value identifying the modification time of the file when it was read. This field shall match the corresponding
 378 value in the `RPMTAG_FILEMTIMES` index in the Header section.

379 *c_filesize*

380 Value identifying the size of the file. This field shall match the corresponding value in the `RPMTAG_FILESIZES`
 381 index in the Header section.

382 *c_devmajor*

383 The major number of the device containing the file system from which the file was read. With the exception of
 384 processing files with `c_nlink > 1`, this field is ignored when installing a package. This field shall match the
 385 corresponding value in the `RPMTAG_FILEDEVICES` index in the Header section.

386 *c_devminor*

387 The minor number of the device containing the file system from which the file was read. With the exception of
 388 processing files with `c_nlink > 1`, this field is ignored when installing a package. This field shall match the
 389 corresponding value in the `RPMTAG_FILEDEVICES` index in the Header section.

390 *c_rdevmajor*

391 The major number of the raw device containing the file system from which the file was read. This field is ignored
 392 when installing a package. This field shall match the corresponding value in the RPMTAG_RDEVS index in the
 393 Header section.

394 *c_rdevminor*

395 The minor number of the raw device containing the file system from which the file was read. This field is ignored
 396 when installing a package. This field shall match the corresponding value in the RPMTAG_RDEVS index in the
 397 Header section.

398 *c_namesize*

399 Value identifying the length of the filename, which is located immediately following the CPIO Header structure.

400 *c_checksum*

401 Value containing the CRC checksum of the file data. This field is not used, and shall contain the value
 402 "00000000". This field is ignored when installing a package.

403 A record with the filename "TRAILER!!!" indicates the last record in the archive.

1.2. Package Script Restrictions

404 Scripts used as part of the package install and uninstall shall only use commands and interfaces that are specified by
 405 the LSB. All other commands are not guaranteed to be present, or to behave in expected ways.

406 Packages shall not use RPM triggers.

407 Packages shall not depend on the order in which scripts are executed (pre-install, pre-uninstall, &c), when doing an
 408 upgrade.

1.3. Package Tools

409 The LSB does not specify the interface to the tools used to manipulate LSB-conformant packages. Each conforming
 410 distribution shall provide documentation for installing LSB packages.

1.4. Package Naming

411 Packages supplied by distributions and applications must follow the following rules for the name field within the
 412 package. These rules are not required for the filename of the package file itself.³

413 The following rules apply to the name field alone, not including any release or version.⁴

- 414 • If the name begins with "lsb-" and contains no other hyphens, the name shall be assigned by the Linux Assigned
 415 Names and Numbers Authority (<http://www.lanana.org>) (LANANA), which shall maintain a registry of LSB names.
 416 The name may be registered by either a distribution or an application.
- 417 • If the package name begins with "lsb-" and contains more than one hyphen (for example
 418 "lsb-distro.example.com-database" or "lsb-gnome-gnumeric"), then the portion of the package name between first
 419 and second hyphens shall either be an LSB provider name assigned by the LANANA, or it may be one of the
 420 owners' fully-qualified domain names in lower case (e.g., "debian.org", "staroffice.sun.com"). The LSB provider

name assigned by LANANA shall only consist of the ASCII characters [a-z0-9]. The provider name or domain name may be either that of a distribution or an application.

- Package names containing no hyphens are reserved for use by distributions. Applications must not use such names.⁵
- Package names which do not start with "lsb-" and which contain a hyphen are open to both distributions and applications. Distributions may name packages in any part of this namespace. They are encouraged to use names from one of the other namespaces available to them, but this is not required due to the large amount of current practice to the contrary.⁶ Applications may name their packages this way, but only if the portion of the name before the first hyphen is a provider name or registered domain name as described above.⁷ Note that package names in this namespace are available to both the distribution and an application. Distributions and applications will need to consider this potential for conflicts when deciding to use these names rather than the alternatives (such as names starting with "lsb-").

1.5. Package Dependencies

Packages shall have a dependency that indicates which LSB modules are required. LSB module descriptions are dash separated tuples containing the name 'lsb', the module name, and the architecture name. The following dependencies may be used.

`lsb-core-arch`

This dependency is used to indicate that the application is dependent on features contained in the LSB-Core specification.

`lsb-core-noarch`

This dependency is used to indicate that the application is dependent on features contained in the LSB-Core specification and that the package does not contain any architecture specific files.

Packages shall not depend on other system-provided dependencies. They shall not depend on non-system-provided dependencies unless those dependencies are fulfilled by packages which are part of the same application. A package may only provide a virtual package name which is registered to that application.

Other modules in the LSB may supplement this list. The architecture specific dependencies are described in the relevant architecture specific LSB.

1.6. Package Architecture Considerations

Packages which do not contain any architecture specific files must specify an architecture of `noarch`. A LSB runtime environment must accept values `noarch`, or the value specified in the architecture specific supplement.

Additional specifications or restrictions may be found in the architecture specific LSB specification.

Notes

1. Supplying an RPM format package is encouraged because it makes systems easier to manage. A future version of the LSB may require RPM, or specify a way for an installer to update a package database.

Applications are also encouraged to uninstall cleanly.

- 453 2. The distribution itself may use a different packaging format for its own packages, and of course it may use any
454 available mechanism for installing the LSB-conformant packages.
- 455 3. For example, there are discrepancies among distributions concerning whether the name might be
456 frobnicator-1.7-21-ppc32.rpm or frobnicator-1.7-21-powerpc32.rpm. The architecture aside, recommended
457 practice is for the filename of the package file to match the name within the package.
- 458 4. For example, if the name with the release and version is frobnicator-1.7-21, the name part is frobnicator and falls
459 under the rules for a name with no hyphens.
- 460 5. For example, "frobnicator".
- 461 6. For example, ssh-common, ssh-client, kernel-pcmcia, and the like. Possible alternative names include sshcommon,
462 foolinux-ssh-common (where foolinux is registered to the distribution), or lsb-foolinux-ssh-common.
- 463 7. For example, if an application vendor has domain name visicalc.example.com and has registered visicalc as a
464 | provider name, they might name packages visicalc-base, visicalc.example.com-charting, and the like.

Free Documentation License

Table of Contents

A. GNU Free Documentation License	1
A.1. PREAMBLE.....	1
A.2. APPLICABILITY AND DEFINITIONS.....	1
A.3. VERBATIM COPYING	2
A.4. COPYING IN QUANTITY	2
A.5. MODIFICATIONS	3
A.6. COMBINING DOCUMENTS	4
A.7. COLLECTIONS OF DOCUMENTS	4
A.8. AGGREGATION WITH INDEPENDENT WORKS.....	4
A.9. TRANSLATION	5
A.10. TERMINATION	5
A.11. FUTURE REVISIONS OF THIS LICENSE	5
A.12. How to use this License for your documents.....	5

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been

designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title Page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.