Linux Standard Base Core Specification 2.0.1

Linux Standard Base Core Specification 2.0.1

Specification Introduction

Specification Introduction

Table of Contents

List of Tables

Foreword

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

Introduction

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

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

I. Introductory Elements

Chapter 1. Scope

1.1. General

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

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

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

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

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

1.2. Module Specific Scope

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

Interfaces described in this module are mandatory except where explicitly listed otherwise. Core interfaces may be 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

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 Languages --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	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-

Name	Title	URL
		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,	http://www.opengroup.org/publicati

Name	**Title**	**URL**
	X/Open Curses, Issue 4, Version 2 (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
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libncurses	libncurses.so.5
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
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

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

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

can

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

cannot

be unable to; there is no possibilty of; it is not possible to

may

is permitted; is allowed; is permissible

need not

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

shall

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

shall not

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

should

it is recommended that; ought to

should not

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

ELF Specification

Table of Contents

List of Tables

List of Figures

I. Low Level System Information

Chapter 1. Operating System Interface

LSB-conforming applications shall assume that stack, heap and other allocated memory regions will be 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 ABI
- System V 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 ABI and the System V 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 restriction may be relaxed in the

Name	Value	Description
		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 values.

Name	Value	Description
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 proceure 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	This section holds a symbol table. 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. Typically, SHT_SYMTAB provides symbols for link editing, though it may also be used for dynamic linking. As a complete symbol table, it

Name	Value	Description
		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

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

4.1.1. ELF Special Sections

The following sections are defined in the System V ABI and the System V ABI Update.

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
.shstrtab	SHT_STRTAB	0

Name	Type	Attributes
.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.

.hash

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

.init

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

.init_array

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

.interp

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

.line

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

.note

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

.preinit_array

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

.rodata

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

.rodata1

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

.shstrtab

This section holds section names.

.strtab

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

.symtab

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

.text

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

4.1.2. Additional Special Sections

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

Table 4-2. Additional Special Sections

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

.ctors

This section contains a list of global constructor function pointers.

.dtors

This section contains a list of global destructor function pointers.

.eh_frame

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

.eh_frame_hdr

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

.gnu.version

This section contains the Symbol Version Table.

.gnu.version_d

This section contains the Version Definitions.

.gnu.version_r

This section contains the Version Requirments.

.jcr

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

.note.ABI-tag

Specify ABI details.

.stab

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

.stabstr

This section contains strings associated with the debugging infomation 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.

Chapter 6. DWARF Extensions

In addition to the Call Frame Instructions defined in section 6.4.2 of DWARF Debugging Information Format, 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) 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 DW_CFA_def_cfa except that the

Name	Value	Meaning
		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.

binary search table

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

7.1. DWARF Exception Header Encoding

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

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.
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.
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.

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 reletive to the current program counter.
DW_EH_PE_datarel	0x30	Value is reletive 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 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;
```

```
        Elfxx_Word    vd_hash;
        Elfxx_Word    vd_aux;
        Elfxx_Word    vd_next;
} Elfxx_Verdef;
```

`vd_version`

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

`vd_flags`

Version information flag bitmask.

`vd_ndx`

Version index numeric value referencing the SHT_GNU_versym section.

`vd_cnt`

Number of associated verdaux array entries.

`vd_hash`

Version name hash value (ELF hash function).

`vd_aux`

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

`vd_next`

Offset to the next verdef entry, in bytes.

Figure 8-2. Version Definition Auxiliary Entries

```
typedef struct {
        Elfxx_Word    vda_name;
        Elfxx_Word    vda_next;
} Elfxx_Verdaux;
```

`vda_name`

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

`vda_next`

Offset to the next verdaux entry, in bytes.

8.3. Version Requirements

Symbol definitions are contained in the special section `.gnu.version_r` which has a section type of `SHT_GNU_verneed`. The number of entries in this section is contained in the `DT_VERNEEDNUM` 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 8-3. 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.

`vna_next`

Offset to the next vernaux entry, in bytes.

8.4. Startup Sequence

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

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

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

8.5. Symbol Resolution

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

Bit number 15 of the version symbol 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.

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

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

Chapter 9. ABI note tag

Every executable shall contain a section named `.note.ABI-tag` of type `SHT_NOTE`. This section is structured as a note section as documented in the ELF spec. The section shall contain at least the following entry. The `name` field (`namesz`/`name`) contains the string "`GNU`". The `type` field shall be 1. The `descsz` field shall be at least 16, and the first 16 bytes of the `desc` field shall be as follows.

The first 32-bit word of the `desc` field shall be 0 (this signifies a Linux executable). The second, third, and fourth 32-bit words of the `desc` field contain the earliest compatible kernel version. For example, if the 3 words are 2, 2, and 5, this signifies a 2.2.5 kernel.

III. Dynamic Linking

Chapter 10. Program Loading and Dynamic Linking

LSB-conforming implementations shall support the object file information and system actions that create running programs as specified in the System V ABI and System V ABI Update and as supplemented by this document and an architecture-specific LSB specification.

Any shared object that is loaded shall contain sufficient DT_NEEDED records to satisfy the symbols on the shared library.

Chapter 11. Program Header

In addition to the Segment Types defined in the System V ABI and System V 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 wether the stack should be executable. The absense of this header indicates 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 ABI and System V ABI 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

DT_PLTRELSZ
: Size in bytes of PLT relocs

DT_REL
: Address of Rel relocs

DT_RELA
: Address of Rela relocs

DT_RELAENT
: Size of one Rela reloc

DT_RELASZ
: Total size of Rela relocs

DT_RELENT
: Size of one Rel reloc

DT_RELSZ
: Total size of Rel relocs

DT_RPATH
: Library search path

DT_SONAME
: Name of shared object

DT_STRSZ
: Size of string table

DT_STRTAB
: Address of string table

DT_SYMBOLIC
: Start symbol search here

DT_SYMENT
: Size of one symbol table entry

DT_SYMTAB
: Address of symbol table

DT_TEXTREL
: Reloc might modify .text

12.1.2. Additional Dynamic Entries

The following dynamic entries are defined here.

DT_ADDRRNGHI

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

DT_ADDRRNGLO

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

DT_AUXILIARY

Shared object to load before self

DT_FILTER

Shared object to get values from

DT_FINI_ARRAY

The address of an array of pointers to termination functions.

DT_FINI_ARRAYSZ

Size in bytes of DT_FINI_ARRAY

DT_HIOS

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

DT_INIT_ARRAY

The address of an array of pointers to initialization functions.

DT_INIT_ARRAYSZ

Size in bytes of DT_INIT_ARRAY

DT_LOOS

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

DT_NUM

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

DT_POSFLAG_1

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

DT_RELCOUNT

All Elf32_Rel R_*_RELATIVE relocations have been placed into a single block and this entry specifies the number of entries in that block. This permits ld.so.1 to streamline the processing of RELATIVE relocations.

DT_SYMINENT

Entry size of syminfo

DT_SYMINFO

Address of the Syminfo table.

DT_SYMINSZ

Size of syminfo table (in bytes)

DT_VALRNGHI

Entries which fall between DT_VALRNGHI & DT_VALRNGLO use the Dyn.d_un.d_val field of the Elf*_Dyn structure.

DT_VALRNGLO

Entries which fall between DT_VALRNGHI & DT_VALRNGLO use the Dyn.d_un.d_val field of the Elf*_Dyn structure.

DT_VERDEF

Address of version definition table

DT_VERDEFNUM

Number of version definitions

DT_VERNEED

Address of table with needed versions

DT_VERNEEDNUM

Number of needed versions

DT_VERSYM

Address of the table provided by the .gnu.version section.

1

Linux Standard Base Specification

Linux Standard Base Specification

Table of Contents

List of Tables

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
this specification
SUSv2
ISO POSIX (2003)
SVID Issue 3
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 [1]	pmap_unset [2]	svcerr_weakauth [3]	xdr_float [3]	xdr_u_char [3]
clnt_create [1]	setdomainname [2]	svctcp_create [2]	xdr_free [3]	xdr_u_int [2]
clnt_pcreateerror [1]	svc_getreqset [3]	svcudp_create [2]	xdr_int [3]	xdr_u_long [3]
clnt_perrno [1]	svc_register [2]	xdr_accepted_reply [3]	xdr_long [3]	xdr_u_short [3]
clnt_perror [1]	svc_run [2]	xdr_array [3]	xdr_opaque [3]	xdr_union [3]
clnt_spcreateerror	svc_sendreply [2]	xdr_bool [3]	xdr_opaque_auth	xdr_vector [3]

[1]			[3]	
clnt_sperrno [1]	svcerr_auth [3]	xdr_bytes [3]	xdr_pointer [3]	xdr_void [3]
clnt_sperror [1]	svcerr_decode [3]	xdr_callhdr [3]	xdr_reference [3]	xdr_wrapstring [3]
getdomainname [2]	svcerr_noproc [3]	xdr_callmsg [3]	xdr_rejected_reply [3]	xdrmem_create [3]
key_decryptsession [3]	svcerr_noprog [3]	xdr_char [3]	xdr_replymsg [3]	xdrrec_create [3]
pmap_getport [2]	svcerr_progvers [3]	xdr_double [3]	xdr_short [3]	xdrrec_eof [3]
pmap_set [2]	svcerr_systemerr [3]	xdr_enum [3]	xdr_string [3]	

Referenced Specification(s)

[1]. SVID Issue 4

[2]. this specification

[3]. SVID Issue 3

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 [1]	fchmod [2]	getwd [2]	read [2]	setrlimit [2]
__getpgid [1]	fchown [2]	initgroups [1]	readdir [2]	setrlimit64 [3]
__lxstat [1]	fcntl [1]	ioctl [1]	readdir_r [2]	setsid [2]
__xmknod [1]	fdatasync [2]	kill [1]	readlink [2]	setuid [2]
__xstat [1]	flock [1]	killpg [2]	readv [2]	sleep [2]
access [2]	fork [2]	lchown [2]	rename [2]	statvfs [2]
acct [1]	fstatvfs [2]	link [2]	rmdir [2]	stime [1]
alarm [2]	fsync [2]	lockf [2]	sbrk [4]	symlink [2]
brk [4]	ftime [2]	lseek [2]	sched_get_priority_max [2]	sync [2]
chdir [2]	ftruncate [2]	mkdir [2]	sched_get_priority_min [2]	sysconf [2]
chmod [2]	getcontext [2]	mkfifo [2]	sched_getparam [2]	time [2]
chown [2]	getegid [2]	mlock [2]	sched_getscheduler	times [2]

			[2]	
chroot [4]	geteuid [2]	mlockall [2]	sched_rr_get_interval [2]	truncate [2]
clock [2]	getgid [2]	mmap [2]	sched_setparam [2]	ulimit [2]
close [2]	getgroups [2]	mprotect [2]	sched_setscheduler [2]	umask [2]
closedir [2]	getitimer [2]	msync [2]	sched_yield [2]	uname [2]
creat [1]	getloadavg [1]	munlock [2]	select [2]	unlink [1]
dup [2]	getpagesize [4]	munlockall [2]	setcontext [2]	utime [2]
dup2 [2]	getpgid [2]	munmap [2]	setegid [2]	utimes [2]
execl [2]	getpgrp [2]	nanosleep [2]	seteuid [2]	vfork [2]
execle [2]	getpid [2]	nice [2]	setgid [2]	wait [2]
execlp [2]	getppid [2]	open [1]	setitimer [2]	wait3 [1]
execv [2]	getpriority [2]	opendir [2]	setpgid [2]	wait4 [1]
execve [2]	getrlimit [2]	pathconf [2]	setpgrp [2]	waitpid [1]
execvp [2]	getrusage [2]	pause [2]	setpriority [2]	write [2]
exit [2]	getsid [2]	pipe [2]	setregid [2]	writev [2]
fchdir [2]	getuid [2]	poll [2]	setreuid [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. Large File Support

[4]. 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 [1]	fgetpos [2]	fsetpos [2]	putchar [2]	sscanf [2]
_IO_getc [1]	fgets [2]	ftell [2]	putchar_unlocked [2]	telldir [2]

_IO_putc [1]	fgetwc_unlocked [1]	ftello [2]	puts [2]	tempnam [2]
_IO_puts [1]	fileno [2]	fwrite [2]	putw [3]	ungetc [2]
asprintf [1]	flockfile [2]	getc [2]	remove [2]	vasprintf [1]
clearerr [2]	fopen [1]	getc_unlocked [2]	rewind [2]	vdprintf [1]
ctermid [2]	fprintf [2]	getchar [2]	rewinddir [2]	vfprintf [2]
fclose [2]	fputc [2]	getchar_unlocked [2]	scanf [2]	vprintf [2]
fdopen [2]	fputs [2]	getw [3]	seekdir [2]	vsnprintf [2]
feof [2]	fread [2]	pclose [2]	setbuf [2]	vsprintf [2]
ferror [2]	freopen [1]	popen [2]	setbuffer [1]	
fflush [2]	fscanf [2]	printf [2]	setvbuf [2]	
fflush_unlocked [1]	fseek [2]	putc [2]	snprintf [2]	
fgetc [2]	fseeko [2]	putc_unlocked [2]	sprintf [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. 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 [1]	stdin [1]	stdout [1]		

Referenced Specification(s)

[1]. ISO POSIX (2003)

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_sigrtmax [1]	sigaddset [2]	sighold [2]	sigpause [2]	sigsuspend [2]
__libc_current_sigrt	sigaltstack [2]	sigignore [2]	sigpending [2]	sigtimedwait [2]

min [1]				
__sigsetjmp [1]	sigandset [1]	siginterrupt [2]	sigprocmask [2]	sigwait [2]
__sysv_signal [1]	sigblock [1]	sigisemptyset [1]	sigqueue [2]	sigwaitinfo [2]
bsd_signal [2]	sigdelset [2]	sigismember [2]	sigrelse [2]	
psignal [1]	sigemptyset [2]	siglongjmp [2]	sigreturn [1]	
raise [2]	sigfillset [2]	signal [2]	sigset [2]	
sigaction [2]	siggetmask [1]	sigorset [1]	sigstack [3]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. 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 [1]				

Referenced Specification(s)

[1]. 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 [1]	catopen [2]	dngettext [1]	iconv_open [2]	setlocale [2]
bindtextdomain [1]	dcgettext [1]	gettext [1]	localeconv [2]	textdomain [1]
catclose [2]	dcngettext [1]	iconv [2]	ngettext [1]	
catgets [2]	dgettext [1]	iconv_close [2]	nl_langinfo [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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 [1]				

Referenced Specification(s)

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

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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

gethostbyname_r [1]				

Referenced Specification(s)

[1]. 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

__wcstod_internal [1]	mbsinit [2]	vwscanf [2]	wcsnlen [1]	wcstoumax [2]
__wcstof_internal [1]	mbsnrtowcs [1]	wcpcpy [1]	wcsnrtombs [1]	wcstouq [1]
__wcstol_internal [1]	mbsrtowcs [2]	wcpncpy [1]	wcspbrk [2]	wcswcs [2]
__wcstold_internal [1]	mbstowcs [2]	wcrtomb [2]	wcsrchr [2]	wcswidth [2]
__wcstoul_internal [1]	mbtowc [2]	wcscasecmp [1]	wcsrtombs [2]	wcsxfrm [2]
btowc [2]	putwc [2]	wcscat [2]	wcsspn [2]	wctob [2]
fgetwc [2]	putwchar [2]	wcschr [2]	wcsstr [2]	wctomb [2]
fgetws [2]	swprintf [2]	wcscmp [2]	wcstod [2]	wctrans [2]
fputwc [2]	swscanf [2]	wcscoll [2]	wcstof [2]	wctype [2]
fputws [2]	towctrans [2]	wcscpy [2]	wcstoimax [2]	wcwidth [2]
fwide [2]	towlower [2]	wcscspn [2]	wcstok [2]	wmemchr [2]
fwprintf [2]	towupper [2]	wcsdup [1]	wcstol [2]	wmemcmp [2]
fwscanf [2]	ungetwc [2]	wcsftime [2]	wcstold [2]	wmemcpy [2]
getwc [2]	vfwprintf [2]	wcslen [2]	wcstoll [2]	wmemmove [2]
getwchar [2]	vfwscanf [2]	wcsncasecmp [1]	wcstombs [2]	wmemset [2]
mblen [2]	vswprintf [2]	wcsncat [2]	wcstoq [1]	wprintf [2]
mbrlen [2]	vswscanf [2]	wcsncmp [2]	wcstoul [2]	wscanf [2]
mbrtowc [2]	vwprintf [2]	wcsncpy [2]	wcstoull [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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 [1]	bzero [2]	strcasestr [1]	strncasecmp [2]	strtoimax [2]
__rawmemchr [1]	ffs [2]	strcat [2]	strncat [2]	strtok [2]
__stpcpy [1]	index [2]	strchr [2]	strncmp [2]	strtok_r [2]
__strdup [1]	memccpy [2]	strcmp [2]	strncpy [2]	strtold [2]
__strtod_internal [1]	memchr [2]	strcoll [2]	strndup [1]	strtoll [2]
__strtof_internal [1]	memcmp [2]	strcpy [2]	strnlen [1]	strtoq [1]
__strtok_r [1]	memcpy [2]	strcspn [2]	strpbrk [2]	strtoull [2]
__strtol_internal [1]	memmove [2]	strdup [2]	strptime [1]	strtoumax [2]
__strtold_internal [1]	memrchr [1]	strerror [2]	strrchr [2]	strtouq [1]
__strtoll_internal [1]	memset [2]	strerror_r [1]	strsep [1]	strverscmp [1]
__strtoul_internal [1]	rindex [2]	strfmon [2]	strsignal [1]	strxfrm [2]
__strtoull_internal [1]	stpcpy [1]	strfry [1]	strspn [2]	swab [2]
bcmp [2]	stpncpy [1]	strftime [2]	strstr [2]	
bcopy [2]	strcasecmp [2]	strlen [2]	strtof [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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 [1]	msgrcv [1]	semget [1]	shmctl [1]	

msgctl [1]	msgsnd [1]	semop [1]	shmdt [1]	
msgget [1]	semctl [1]	shmat [1]	shmget [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

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 [1]	regerror [1]	regexec [1]	regfree [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

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 [1]	re_comp [1]	re_exec [1]	step [1]	

Referenced Specification(s)

[1]. 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 [1]	loc2 [1]	locs [1]		

Referenced Specification(s)

[1]. 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) [1]	isalpha [2]	ispunct [2]	iswctype [2]	iswupper [2]
__ctype_get_mb_cur_max [1]	isascii [2]	isspace [2]	iswdigit [2]	iswxdigit [2]
__ctype_tolower_loc(GLIBC_2.3) [1]	iscntrl [2]	isupper [2]	iswgraph [2]	isxdigit [2]
__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]. this specification

[2]. ISO POSIX (2003)

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]	ctime [2]	gmtime [2]	localtime_r [2]	ualarm [2]
asctime [2]	ctime_r [2]	gmtime_r [2]	mktime [2]	
asctime_r [2]	difftime [2]	localtime [2]	tzset [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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]. 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 [1]	__tzname [1]	timezone [2]		
__timezone [1]	daylight [2]	tzname [2]		

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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

cfgetispeed [1]	cfsetispeed [1]	tcdrain [1]	tcgetattr [1]	tcsendbreak [1]
cfgetospeed [1]	cfsetospeed [1]	tcflow [1]	tcgetpgrp [1]	tcsetattr [1]
cfmakeraw [2]	cfsetspeed [2]	tcflush [1]	tcgetsid [1]	tcsetpgrp [1]

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this 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 [1]	getgrgid [1]	getprotobynumber [1]	getservbyport [1]	setgrent [1]
endnetent [1]	getgrgid_r [1]	getprotoent [1]	getservent [1]	setgroups [2]
endprotoent [1]	getgrnam [1]	getpwent [1]	getutent [2]	setnetent [1]
endpwent [1]	getgrnam_r [1]	getpwnam [1]	getutent_r [2]	setprotoent [1]
endservent [1]	gethostbyaddr [1]	getpwnam_r [1]	getutxent [1]	setpwent [1]
endutent [3]	gethostbyname [1]	getpwuid [1]	getutxid [1]	setservent [1]
endutxent [1]	getnetbyaddr [1]	getpwuid_r [1]	getutxline [1]	setutent [2]
getgrent [1]	getprotobyname [1]	getservbyname [1]	pututxline [1]	setutxent [1]

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

[3]. 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 [1]	__register_atfork(GLIBC_2.3.2) [1]	_obstack_begin [1]	_obstack_newchunk [1]	obstack_free [1]

Referenced Specification(s)

[1]. 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 [1]	fopen64 [2]	ftello64 [2]	lseek64 [2]	readdir64 [2]
__lxstat64 [1]	freopen64 [2]	ftruncate64 [2]	mkstemp64 [2]	statvfs64 [2]

__xstat64 [1]	fseeko64 [2]	ftw64 [2]	mmap64 [2]	tmpfile64 [2]
creat64 [2]	fsetpos64 [2]	getrlimit64 [2]	nftw64 [2]	truncate64 [2]
fgetpos64 [2]	fstatvfs64 [2]	lockf64 [2]	open64 [2]	

Referenced Specification(s)

[1]. this specification

[2]. Large File Support

1.2.17. Standard Library

1.2.17.1. Interfaces for Standard Library

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

Table 1-26. libc - Standard Library Function Interfaces

_Exit [1]	dirname [1]	glob [1]	lsearch [1]	srand [1]
__assert_fail [2]	div [1]	glob64 [2]	makecontext [1]	srand48 [1]
__cxa_atexit [2]	drand48 [1]	globfree [1]	malloc [1]	srandom [1]
__errno_location [2]	ecvt [1]	globfree64 [2]	memmem [2]	strtod [1]
__fpending [2]	erand48 [1]	grantpt [1]	mkstemp [1]	strtol [1]
__getpagesize [2]	err [2]	hcreate [1]	mktemp [1]	strtoul [1]
__isinf [2]	error [2]	hdestroy [1]	mrand48 [1]	swapcontext [1]
__isinff [2]	errx [2]	hsearch [1]	nftw [1]	syslog [1]
__isinfl [2]	fcvt [1]	htonl [1]	nrand48 [1]	system [2]
__isnan [2]	fmtmsg [1]	htons [1]	ntohl [1]	tdelete [1]
__isnanf [2]	fnmatch [1]	imaxabs [1]	ntohs [1]	tfind [1]
__isnanl [2]	fpathconf [1]	imaxdiv [1]	openlog [1]	tmpfile [1]
__sysconf [2]	free [1]	inet_addr [1]	perror [1]	tmpnam [1]
_exit [1]	freeaddrinfo [1]	inet_ntoa [1]	posix_memalign [1]	tsearch [1]
_longjmp [1]	ftrylockfile [1]	inet_ntop [1]	ptsname [1]	ttyname [1]
_setjmp [1]	ftw [1]	inet_pton [1]	putenv [1]	ttyname_r [1]
a64l [1]	funlockfile [1]	initstate [1]	qsort [1]	twalk [1]
abort [1]	gai_strerror [1]	insque [1]	rand [1]	unlockpt [1]
abs [1]	gcvt [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]	vfscanf [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]
closelog [1]	getopt_long [2]	llabs [1]	sethostid [2]	warnx [2]
confstr [1]	getopt_long_only [2]	lldiv [1]	sethostname [2]	wordexp [1]
cuserid [3]	getsubopt [1]	longjmp [1]	setlogmask [1]	wordfree [1]
daemon [2]	gettimeofday [1]	lrand48 [1]	setstate [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

[3]. 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 [1]	environ [2]	optarg [2]	optind [1]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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;
  unsigned char d_type;
  char d_name[256];
}
 ;
struct dirent64
{
  uint64_t d_ino;
  int64_t d_off;
  unsigned short d_reclen;
  unsigned char d_type;
  char d_name[256];
}
 ;
```

1.3.4. errno.h

```
#define errno   (*__errno_location())

#define EPERM   1
#define ECHILD  10
#define ENETDOWN        100
```

```
#define ENETUNREACH     101
#define ENETRESET       102
#define ECONNABORTED    103
#define ECONNRESET      104
#define ENOBUFS 105
#define EISCONN 106
#define ENOTCONN        107
#define ESHUTDOWN       108
#define ETOOMANYREFS    109
#define EAGAIN  11
#define ETIMEDOUT       110
#define ECONNREFUSED    111
#define EHOSTDOWN       112
#define EHOSTUNREACH    113
#define EALREADY        114
#define EINPROGRESS     115
#define ESTALE  116
#define EUCLEAN 117
#define ENOTNAM 118
#define ENAVAIL 119
#define ENOMEM  12
#define EISNAM  120
#define EREMOTEIO       121
#define EDQUOT  122
#define ENOMEDIUM       123
#define EMEDIUMTYPE     124
#define ECANCELED       125
#define EACCES  13
#define EFAULT  14
#define ENOTBLK 15
#define EBUSY   16
#define EEXIST  17
#define EXDEV   18
#define ENODEV  19
#define ENOENT  2
#define ENOTDIR 20
#define EISDIR  21
#define EINVAL  22
#define ENFILE  23
#define EMFILE  24
#define ENOTTY  25
#define ETXTBSY 26
#define EFBIG   27
#define ENOSPC  28
#define ESPIPE  29
#define ESRCH   3
#define EROFS   30
#define EMLINK  31
#define EPIPE   32
#define EDOM    33
#define ERANGE  34
#define EDEADLK 35
#define ENAMETOOLONG    36
```

```
312   #define ENOLCK  37
313   #define ENOSYS  38
314   #define ENOTEMPTY       39
315   #define EINTR   4
316   #define ELOOP   40
317   #define ENOMSG  42
318   #define EIDRM   43
319   #define ECHRNG  44
320   #define EL2NSYNC        45
321   #define EL3HLT  46
322   #define EL3RST  47
323   #define ELNRNG  48
324   #define EUNATCH 49
325   #define EIO     5
326   #define ENOANO  55
327   #define EBADRQC 56
328   #define EBADSLT 57
329   #define EBFONT  59
330   #define ENXIO   6
331   #define ENOSTR  60
332   #define ENODATA 61
333   #define ETIME   62
334   #define ENOSR   63
335   #define ENONET  64
336   #define ENOPKG  65
337   #define EREMOTE 66
338   #define ENOLINK 67
339   #define EADV    68
340   #define ESRMNT  69
341   #define E2BIG   7
342   #define ECOMM   70
343   #define EPROTO  71
344   #define EMULTIHOP       72
345   #define EDOTDOT 73
346   #define EBADMSG 74
347   #define EOVERFLOW       75
348   #define ENOTUNIQ        76
349   #define EBADFD  77
350   #define EREMCHG 78
351   #define ELIBACC 79
352   #define ENOEXEC 8
353   #define ELIBBAD 80
354   #define ELIBSCN 81
355   #define ELIBMAX 82
356   #define ELIBEXEC        83
357   #define EILSEQ  84
358   #define ERESTART        85
359   #define ESTRPIPE        86
360   #define EUSERS  87
361   #define ENOTSOCK        88
362   #define EDESTADDRREQ    89
363   #define EBADF   9
364   #define EMSGSIZE        90
```

```
#define EPROTOTYPE      91
#define ENOPROTOOPT     92
#define EPROTONOSUPPORT 93
#define ESOCKTNOSUPPORT 94
#define EOPNOTSUPP      95
#define EPFNOSUPPORT    96
#define EAFNOSUPPORT    97
#define EADDRINUSE      98
#define EADDRNOTAVAIL   99
#define EWOULDBLOCK     EAGAIN
#define ENOTSUP EOPNOTSUPP
```

1.3.5. fcntl.h

```

#define O_RDONLY        00
#define O_ACCMODE       0003
#define O_WRONLY        01
#define O_CREAT 0100
#define O_TRUNC 01000
#define O_SYNC  010000
#define O_RDWR  02
#define O_EXCL  0200
#define O_APPEND        02000
#define O_ASYNC 020000
#define O_NOCTTY        0400
#define O_NDELAY        04000
#define O_NONBLOCK      04000
#define FD_CLOEXEC      1

struct flock
{
  short l_type;
  short l_whence;
  off_t l_start;
  off_t l_len;
  pid_t l_pid;
}
 ;
struct flock64
{
  short l_type;
  short l_whence;
  loff_t l_start;
  loff_t l_len;
  pid_t l_pid;
}
 ;

#define F_DUPFD 0
#define F_RDLCK 0
#define F_GETFD 1
```

```
#define F_WRLCK 1
#define F_SETFD 2
#define F_UNLCK 2
#define F_GETFL 3
#define F_SETFL 4
#define F_GETLK 5
#define F_SETLK 6
#define F_SETLKW        7
#define F_SETOWN        8
#define F_GETOWN        9
```

1.3.6. fmtmsg.h

```

#define MM_HARD 1
#define MM_NRECOV       128
#define MM_UTIL 16
#define MM_SOFT 2
#define MM_OPSYS        32
#define MM_FIRM 4
#define MM_RECOVER      64
#define MM_APPL 8

#define MM_NOSEV        0
#define MM_HALT 1
#define MM_ERROR        2

#define MM_NULLLBL      ((char *) 0)
```

1.3.7. fnmatch.h

```

#define FNM_PATHNAME    (1<<0)
#define FNM_NOESCAPE    (1<<1)
#define FNM_PERIOD      (1<<2)
#define FNM_NOMATCH     1
```

1.3.8. ftw.h

```

#define FTW_D   FTW_D
#define FTW_DNR FTW_DNR
#define FTW_DP  FTW_DP
#define FTW_F   FTW_F
#define FTW_NS  FTW_NS
#define FTW_SL  FTW_SL
#define FTW_SLN FTW_SLN

enum
{
  FTW_F, FTW_D, FTW_DNR, FTW_NS, FTW_SL, FTW_DP, FTW_SLN
}
```

```
 ;

enum
{
  FTW_PHYS, FTW_MOUNT, FTW_CHDIR, FTW_DEPTH
}
 ;

struct FTW
{
  int base;
  int level;
}
 ;

typedef int (*__ftw_func_t) (char *__filename, struct stat * __status,
                             int __flag);
typedef int (*__ftw64_func_t) (char *__filename, struct stat64 * __status,
                               int __flag);
typedef int (*__nftw_func_t) (char *__filename, struct stat * __status,
                              int __flag, struct FTW * __info);
typedef int (*__nftw64_func_t) (char *__filename, struct stat64 * __status,
                                int __flag, struct FTW * __info);
```

1.3.9. getopt.h

```

#define no_argument     0
#define required_argument       1
#define optional_argument       2

struct option
{
  char *name;
  int has_arg;
  int *flag;
  int val;
}
 ;
```

1.3.10. glob.h

```

#define GLOB_ERR        (1<<0)
#define GLOB_MARK       (1<<1)
#define GLOB_BRACE      (1<<10)
#define GLOB_NOMAGIC    (1<<11)
#define GLOB_TILDE      (1<<12)
#define GLOB_ONLYDIR    (1<<13)
#define GLOB_TILDE_CHECK        (1<<14)
#define GLOB_NOSORT     (1<<2)
#define GLOB_DOOFFS     (1<<3)
```

```
#define GLOB_NOCHECK    (1<<4)
#define GLOB_APPEND     (1<<5)
#define GLOB_NOESCAPE   (1<<6)
#define GLOB_PERIOD     (1<<7)
#define GLOB_MAGCHAR    (1<<8)
#define GLOB_ALTDIRFUNC (1<<9)

#define GLOB_NOSPACE    1
#define GLOB_ABORTED    2
#define GLOB_NOMATCH    3
#define GLOB_NOSYS      4

typedef struct
{
  size_t gl_pathc;
  char **gl_pathv;
  size_t gl_offs;
  int gl_flags;
  void (*gl_closedir) (void *);
  struct dirent *(*gl_readdir) (void *);
  void *(*gl_opendir) (const char *);
  int (*gl_lstat) (const char *, struct stat *);
  int (*gl_stat) (const char *, struct stat *);
}
glob_t;

typedef struct
{
  size_t gl_pathc;
  char **gl_pathv;
  size_t gl_offs;
  int gl_flags;
  void (*gl_closedir) (void *);
  struct dirent64 *(*gl_readdir64) (void *);
  void *(*gl_opendir) (const char *);
  int (*gl_lstat) (const char *, struct stat *);
  int (*gl_stat) (const char *, struct stat *);
}
glob64_t;
```

1.3.11. grp.h

```

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

1.3.12. iconv.h

```
551
552 typedef void *iconv_t;
```

1.3.13. inttypes.h

```
553
554 typedef lldiv_t imaxdiv_t;
555 typedef unsigned char uint8_t;
556 typedef unsigned short uint16_t;
557 typedef unsigned int uint32_t;
```

1.3.14. langinfo.h

```
558
559 #define ABDAY_1 0x20000
560 #define ABDAY_2 0x20001
561 #define ABDAY_3 0x20002
562 #define ABDAY_4 0x20003
563 #define ABDAY_5 0x20004
564 #define ABDAY_6 0x20005
565 #define ABDAY_7 0x20006
566
567 #define DAY_1   0x20007
568 #define DAY_2   0x20008
569 #define DAY_3   0x20009
570 #define DAY_4   0x2000A
571 #define DAY_5   0x2000B
572 #define DAY_6   0x2000C
573 #define DAY_7   0x2000D
574
575 #define ABMON_1 0x2000E
576 #define ABMON_2 0x2000F
577 #define ABMON_3 0x20010
578 #define ABMON_4 0x20011
579 #define ABMON_5 0x20012
580 #define ABMON_6 0x20013
581 #define ABMON_7 0x20014
582 #define ABMON_8 0x20015
583 #define ABMON_9 0x20016
584 #define ABMON_10        0x20017
585 #define ABMON_11        0x20018
586 #define ABMON_12        0x20019
587
588 #define MON_1   0x2001A
589 #define MON_2   0x2001B
590 #define MON_3   0x2001C
591 #define MON_4   0x2001D
592 #define MON_5   0x2001E
593 #define MON_6   0x2001F
594 #define MON_7   0x20020
```

```
595  #define MON_8   0x20021
596  #define MON_9   0x20022
597  #define MON_10  0x20023
598  #define MON_11  0x20024
599  #define MON_12  0x20025
600
601  #define AM_STR  0x20026
602  #define PM_STR  0x20027
603
604  #define D_T_FMT 0x20028
605  #define D_FMT   0x20029
606  #define T_FMT   0x2002A
607  #define T_FMT_AMPM      0x2002B
608
609  #define ERA     0x2002C
610  #define ERA_D_FMT       0x2002E
611  #define ALT_DIGITS      0x2002F
612  #define ERA_D_T_FMT     0x20030
613  #define ERA_T_FMT       0x20031
614
615  #define CODESET 14
616
617  #define CRNCYSTR        0x4000F
618
619  #define RADIXCHAR       0x10000
620  #define THOUSEP 0x10001
621  #define YESEXPR 0x50000
622  #define NOEXPR  0x50001
623  #define YESSTR  0x50002
624  #define NOSTR   0x50003
```

1.3.15. limits.h

```
625
626  #define LLONG_MIN       (-LLONG_MAX-1LL)
627  #define ULLONG_MAX      18446744073709551615ULL
628  #define OPEN_MAX        256
629  #define PATH_MAX        4096
630  #define LLONG_MAX       9223372036854775807LL
631  #define SSIZE_MAX       LONG_MAX
632
633  #define MB_LEN_MAX      16
634
635  #define SCHAR_MIN       (-128)
636  #define SCHAR_MAX       127
637  #define UCHAR_MAX       255
638  #define CHAR_BIT        8
639
640  #define SHRT_MIN        (-32768)
641  #define SHRT_MAX        32767
642  #define USHRT_MAX       65535
643
```

```
#define INT_MIN (-INT_MAX-1)
#define INT_MAX 2147483647
#define __INT_MAX__     2147483647
#define UINT_MAX        4294967295U

#define LONG_MIN        (-LONG_MAX-1L)
```

1.3.16. locale.h

```

#define LC_CTYPE        0
#define LC_NUMERIC      1
#define LC_TELEPHONE    10
#define LC_MEASUREMENT  11
#define LC_IDENTIFICATION       12
#define LC_TIME 2
#define LC_COLLATE      3
#define LC_MONETARY     4
#define LC_MESSAGES     5
#define LC_ALL  6
#define LC_PAPER        7
#define LC_NAME 8
#define LC_ADDRESS      9

struct lconv
{
  char *decimal_point;
  char *thousands_sep;
  char *grouping;
  char *int_curr_symbol;
  char *currency_symbol;
  char *mon_decimal_point;
  char *mon_thousands_sep;
  char *mon_grouping;
  char *positive_sign;
  char *negative_sign;
  char int_frac_digits;
  char frac_digits;
  char p_cs_precedes;
  char p_sep_by_space;
  char n_cs_precedes;
  char n_sep_by_space;
  char p_sign_posn;
  char n_sign_posn;
  char int_p_cs_precedes;
  char int_p_sep_by_space;
  char int_n_cs_precedes;
  char int_n_sep_by_space;
  char int_p_sign_posn;
  char int_n_sign_posn;
}
 ;
```

```
693
694   typedef struct __locale_struct
695   {
696     struct locale_data *__locales[13];
697     const unsigned short *__ctype_b;
698     const int *__ctype_tolower;
699     const int *__ctype_toupper;
700     const char *__names[13];
701   }
702    *__locale_t;
```

1.3.17. net/if.h

```
703
704   #define IF_NAMESIZE     16
705
706   #define IFF_UP  0x01
707   #define IFF_BROADCAST   0x02
708   #define IFF_DEBUG       0x04
709   #define IFF_LOOPBACK    0x08
710   #define IFF_POINTOPOINT 0x10
711   #define IFF_PROMISC     0x100
712   #define IFF_MULTICAST   0x1000
713   #define IFF_NOTRAILERS  0x20
714   #define IFF_RUNNING     0x40
715   #define IFF_NOARP       0x80
716
717   struct ifaddr
718   {
719     struct sockaddr ifa_addr;
720     union
721     {
722       struct sockaddr ifu_broadaddr;
723       struct sockaddr ifu_dstaddr;
724     }
725     ifa_ifu;
726     void *ifa_ifp;
727     void *ifa_next;
728   }
729    ;
730   #define IFNAMSIZ        IF_NAMESIZE
731
732   struct ifreq
733   {
734     union
735     {
736       char ifrn_name[IFNAMSIZ];
737     }
738     ifr_ifrn;
739     union
740     {
741       struct sockaddr ifru_addr;
```

```
      struct sockaddr ifru_dstaddr;
      struct sockaddr ifru_broadaddr;
      struct sockaddr ifru_netmask;
      struct sockaddr ifru_hwaddr;
      short ifru_flags;
      int ifru_ivalue;
      int ifru_mtu;
      char ifru_slave[IFNAMSIZ];
      char ifru_newname[IFNAMSIZ];
      caddr_t ifru_data;
      struct ifmap ifru_map;
    }
    ifr_ifru;
  }
   ;

  struct ifconf
  {
    int ifc_len;
    union
    {
      caddr_t ifcu_buf;
      struct ifreq *ifcu_req;
    }
    ifc_ifcu;
  }
   ;
```

1.3.18. netdb.h

```

  #define h_errno (*__h_errno_location ())
  #define NETDB_INTERNAL  -1
  #define NETDB_SUCCESS   0
  #define HOST_NOT_FOUND  1
  #define IPPORT_RESERVED 1024
  #define NI_MAXHOST      1025
  #define TRY_AGAIN       2
  #define NO_RECOVERY     3
  #define NI_MAXSERV      32
  #define NO_DATA 4
  #define h_addr  h_addr_list[0]
  #define NO_ADDRESS      NO_DATA

  struct servent
  {
    char *s_name;
    char **s_aliases;
    int s_port;
    char *s_proto;
  }
   ;
```

```
791   struct hostent
792   {
793     char *h_name;
794     char **h_aliases;
795     int h_addrtype;
796     int h_length;
797     char **h_addr_list;
798   }
799    ;
800   struct protoent
801   {
802     char *p_name;
803     char **p_aliases;
804     int p_proto;
805   }
806    ;
807   struct netent
808   {
809     char *n_name;
810     char **n_aliases;
811     int n_addrtype;
812     unsigned int n_net;
813   }
814    ;
815   #define AI_PASSIVE      0x0001
816   #define AI_CANONNAME    0x0002
817   #define AI_NUMERICHOST  0x0004
818
819   struct addrinfo
820   {
821     int ai_flags;
822     int ai_family;
823     int ai_socktype;
824     int ai_protocol;
825     socklen_t ai_addrlen;
826     struct sockaddr *ai_addr;
827     char *ai_canonname;
828     struct addrinfo *ai_next;
829   }
830    ;
831   #define NI_NUMERICHOST  1
832   #define NI_DGRAM        16
833   #define NI_NUMERICSERV  2
834   #define NI_NOFQDN       4
835   #define NI_NAMEREQD     8
836
837   #define EAI_BADFLAGS    -1
838   #define EAI_MEMORY      -10
839   #define EAI_SYSTEM      -11
840   #define EAI_NONAME      -2
841   #define EAI_AGAIN       -3
842   #define EAI_FAIL        -4
843   #define EAI_NODATA      -5
```

```
#define EAI_FAMILY      -6
#define EAI_SOCKTYPE    -7
#define EAI_SERVICE     -8
#define EAI_ADDRFAMILY  -9
```

1.3.19. netinet/in.h

```

#define IPPROTO_IP      0
#define IPPROTO_ICMP    1
#define IPPROTO_UDP     17
#define IPPROTO_IGMP    2
#define IPPROTO_RAW     255
#define IPPROTO_IPV6    41
#define IPPROTO_ICMPV6  58
#define IPPROTO_TCP     6

typedef uint16_t in_port_t;

struct in_addr
{
  uint32_t s_addr;
}
 ;
typedef uint32_t in_addr_t;
#define INADDR_NONE     ((in_addr_t) 0xffffffff)
#define INADDR_BROADCAST        (0xffffffff)
#define INADDR_ANY      0

struct in6_addr
{
  union
  {
    uint8_t u6_addr8[16];
    uint16_t u6_addr16[8];
    uint32_t u6_addr32[4];
  }
  in6_u;
}
 ;
#define IN6ADDR_ANY_INIT        { { { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 } } }
#define IN6ADDR_LOOPBACK_INIT   { { { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 } } }

#define INET_ADDRSTRLEN 16

struct sockaddr_in
{
  sa_family_t sin_family;
  unsigned short sin_port;
  struct in_addr sin_addr;
  unsigned char sin_zero[8];
}
```

```
 ;
#define INET6_ADDRSTRLEN        46

struct sockaddr_in6
{
  unsigned short sin6_family;
  uint16_t sin6_port;
  uint32_t sin6_flowinfo;
  struct in6_addr sin6_addr;
  uint32_t sin6_scope_id;
}
 ;
#define SOL_IP  0
#define IP_TOS  1
#define IPV6_UNICAST_HOPS       16
#define IPV6_MULTICAST_IF       17
#define IPV6_MULTICAST_HOPS     18
#define IPV6_MULTICAST_LOOP     19
#define IPV6_JOIN_GROUP 20
#define IPV6_LEAVE_GROUP        21
#define IPV6_V6ONLY     26
#define IP_MULTICAST_IF 32
#define IP_MULTICAST_TTL        33
#define IP_MULTICAST_LOOP       34
#define IP_ADD_MEMBERSHIP       35
#define IP_DROP_MEMBERSHIP      36

struct ipv6_mreq
{
  struct in6_addr ipv6mr_multiaddr;
  int ipv6mr_interface;
}
 ;
struct ip_mreq
{
  struct in_addr imr_multiaddr;
  struct in_addr imr_interface;
}
 ;
```

1.3.20. netinet/tcp.h

```

#define TCP_NODELAY     1
#define SOL_TCP 6
```

1.3.21. netinet/udp.h

```

#define SOL_UDP 17
```

1.3.22. nl_types.h

```

#define NL_CAT_LOCALE   1
#define NL_SETD 1

typedef void *nl_catd;

typedef int nl_item;
```

1.3.23. pty.h

```

struct winsize
{
  unsigned short ws_row;
  unsigned short ws_col;
  unsigned short ws_xpixel;
  unsigned short ws_ypixel;
}
 ;
```

1.3.24. pwd.h

```

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

1.3.25. regex.h

```

#define RE_BACKSLASH_ESCAPE_IN_LISTS    ((unsigned long int)1)
#define RE_BK_PLUS_QM   (RE_BACKSLASH_ESCAPE_IN_LISTS<<1)
#define RE_SYNTAX_AWK   (RE_BACKSLASH_ESCAPE_IN_LISTS|RE_DOT_NOT_NULL|RE_NO_BK_PARENS|
RE_NO_BK_REFS| RE_NO_BK_VBAR| RE_NO_EMPTY_RANGES| RE_DOT_NEWLINE|
RE_CONTEXT_INDEP_ANCHORS| RE_UNMATCHED_RIGHT_PAREN_ORD | RE_NO_GNU_OPS)
#define RE_CHAR_CLASSES (RE_BK_PLUS_QM<<1)
#define RE_SYNTAX_GREP
(RE_BK_PLUS_QM|RE_CHAR_CLASSES|RE_HAT_LISTS_NOT_NEWLINE|RE_INTERVALS|RE_NEWLINE_ALT)
#define RE_CONTEXT_INDEP_ANCHORS        (RE_CHAR_CLASSES<<1)
```

```
975  #define RE_SYNTAX_EGREP (RE_CHAR_CLASSES|RE_CONTEXT_INDEP_ANCHORS|
976  RE_CONTEXT_INDEP_OPS|RE_HAT_LISTS_NOT_NEWLINE|RE_NEWLINE_ALT|RE_NO_BK_PARENS|RE_NO_BK_
977  VBAR)
978  #define _RE_SYNTAX_POSIX_COMMON
979  (RE_CHAR_CLASSES|RE_DOT_NEWLINE|RE_DOT_NOT_NULL|RE_INTERVALS|RE_NO_EMPTY_RANGES)
980  #define RE_CONTEXT_INDEP_OPS    (RE_CONTEXT_INDEP_ANCHORS<<1)
981  #define RE_CONTEXT_INVALID_OPS  (RE_CONTEXT_INDEP_OPS<<1)
982  #define RE_DOT_NEWLINE  (RE_CONTEXT_INVALID_OPS<<1)
983  #define RE_INVALID_INTERVAL_ORD (RE_DEBUG<<1)
984  #define RE_DOT_NOT_NULL (RE_DOT_NEWLINE<<1)
985  #define RE_HAT_LISTS_NOT_NEWLINE        (RE_DOT_NOT_NULL<<1)
986  #define RE_INTERVALS    (RE_HAT_LISTS_NOT_NEWLINE<<1)
987  #define RE_LIMITED_OPS  (RE_INTERVALS<<1)
988  #define RE_NEWLINE_ALT  (RE_LIMITED_OPS<<1)
989  #define RE_NO_BK_BRACES (RE_NEWLINE_ALT<<1)
990  #define RE_NO_BK_PARENS (RE_NO_BK_BRACES<<1)
991  #define RE_NO_BK_REFS   (RE_NO_BK_PARENS<<1)
992  #define RE_NO_BK_VBAR   (RE_NO_BK_REFS<<1)
993  #define RE_NO_EMPTY_RANGES      (RE_NO_BK_VBAR<<1)
994  #define RE_UNMATCHED_RIGHT_PAREN_ORD    (RE_NO_EMPTY_RANGES<<1)
995  #define RE_DEBUG        (RE_NO_GNU_OPS<<1)
996  #define RE_NO_GNU_OPS   (RE_NO_POSIX_BACKTRACKING<<1)
997  #define RE_SYNTAX_POSIX_EGREP
998  (RE_SYNTAX_EGREP|RE_INTERVALS|RE_NO_BK_BRACES|RE_INVALID_INTERVAL_ORD)
999  #define RE_SYNTAX_POSIX_AWK
1000 (RE_SYNTAX_POSIX_EXTENDED|RE_BACKSLASH_ESCAPE_IN_LISTS|RE_INTERVALS|RE_NO_GNU_OPS)
1001 #define RE_NO_POSIX_BACKTRACKING        (RE_UNMATCHED_RIGHT_PAREN_ORD<<1)
1002 #define RE_SYNTAX_POSIX_BASIC   (_RE_SYNTAX_POSIX_COMMON|RE_BK_PLUS_QM)
1003 #define RE_SYNTAX_POSIX_EXTENDED
1004 (_RE_SYNTAX_POSIX_COMMON|RE_CONTEXT_INDEP_ANCHORS|RE_CONTEXT_INDEP_OPS|RE_NO_BK_BRACES
1005 |RE_NO_BK_PARENS|RE_NO_BK_VBAR|RE_CONTEXT_INVALID_OPS|RE_UNMATCHED_RIGHT_PAREN_ORD)
1006 #define RE_SYNTAX_POSIX_MINIMAL_EXTENDED
1007 (_RE_SYNTAX_POSIX_COMMON|RE_CONTEXT_INDEP_ANCHORS|RE_CONTEXT_INVALID_OPS|RE_NO_BK_BRAC
1008 ES|RE_NO_BK_PARENS|RE_NO_BK_REFS|RE_NO_BK_VBAR|RE_UNMATCHED_RIGHT_PAREN_ORD)
1009 #define RE_SYNTAX_POSIX_MINIMAL_BASIC   (_RE_SYNTAX_POSIX_COMMON|RE_LIMITED_OPS)
1010 #define RE_SYNTAX_ED    RE_SYNTAX_POSIX_BASIC
1011 #define RE_SYNTAX_SED   RE_SYNTAX_POSIX_BASIC
1012
1013 typedef unsigned long reg_syntax_t;
1014
1015 typedef struct re_pattern_buffer
1016 {
1017   unsigned char *buffer;
1018   unsigned long allocated;
1019   unsigned long used;
1020   reg_syntax_t syntax;
1021   char *fastmap;
1022   char *translate;
1023   size_t re_nsub;
1024   unsigned int can_be_null:1;
1025   unsigned int regs_allocated:2;
1026   unsigned int fastmap_accurate:1;
1027   unsigned int no_sub:1;
```

```
  unsigned int not_bol:1;
  unsigned int not_eol:1;
  unsigned int newline_anchor:1;
}
regex_t;
typedef int regoff_t;
typedef struct
{
  regoff_t rm_so;
  regoff_t rm_eo;
}
regmatch_t;
#define REG_NOTEOL      (1<<1)
#define REG_ICASE       (REG_EXTENDED<<1)
#define REG_NEWLINE     (REG_ICASE<<1)
#define REG_NOSUB       (REG_NEWLINE<<1)
#define REG_NOMATCH     -1
#define REG_EXTENDED    1
#define REG_NOTBOL      1
```

1.3.26. rpc/auth.h

```

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

union des_block
{
  struct
  {
    u_int32_t high;
    u_int32_t low;
  }
  key;
  char c[8];
}
 ;

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

typedef struct AUTH
```

```
{
  struct opaque_auth ah_cred;
  struct opaque_auth ah_verf;
  union des_block ah_key;
  struct auth_ops *ah_ops;
  caddr_t ah_private;
}
AUTH;

struct auth_ops
{
  void (*ah_nextverf) (struct AUTH *);
  int (*ah_marshal) (struct AUTH *, XDR *);
  int (*ah_validate) (struct AUTH *, struct opaque_auth *);
  int (*ah_refresh) (struct AUTH *);
  void (*ah_destroy) (struct AUTH *);
}
 ;
```

1.3.27. rpc/clnt.h

```

#define clnt_control(cl,rq,in)  ((*(cl)->cl_ops->cl_control)(cl,rq,in))
#define clnt_abort(rh)  ((*(rh)->cl_ops->cl_abort)(rh))
#define clnt_call(rh, proc, xargs, argsp, xres, resp, secs)   ((*(rh)->cl_ops->cl_call)(rh,
proc, xargs, argsp, xres, resp, secs))
#define clnt_destroy(rh)        ((*(rh)->cl_ops->cl_destroy)(rh))
#define clnt_freeres(rh,xres,resp)      ((*(rh)->cl_ops->cl_freeres)(rh,xres,resp))
#define clnt_geterr(rh,errp)    ((*(rh)->cl_ops->cl_geterr)(rh, errp))
#define NULLPROC        ((u_long)0)
#define CLSET_TIMEOUT   1
#define CLGET_XID       10
#define CLSET_XID       11
#define CLGET_VERS      12
#define CLSET_VERS      13
#define CLGET_PROG      14
#define CLSET_PROG      15
#define CLGET_TIMEOUT   2
#define CLGET_SERVER_ADDR       3
#define CLSET_RETRY_TIMEOUT     4
#define CLGET_RETRY_TIMEOUT     5
#define CLGET_FD        6
#define CLGET_SVC_ADDR  7
#define CLSET_FD_CLOSE  8
#define CLSET_FD_NCLOSE 9

enum clnt_stat
{
  RPC_SUCCESS, RPC_CANTENCODEARGS = 1, RPC_CANTDECODERES = 2, RPC_CANTSEND =
    3, RPC_CANTRECV = 4, RPC_TIMEDOUT = 5, RPC_VERSMISMATCH =
    6, RPC_AUTHERROR = 7, RPC_PROGUNAVAIL = 8, RPC_PROGVERSMISMATCH =
    9, RPC_PROCUNAVAIL = 10, RPC_CANTDECODEARGS = 11, RPC_SYSTEMERROR =
```

```
1126        12, RPC_NOBROADCAST = 21, RPC_UNKNOWNHOST = 13, RPC_UNKNOWNPROTO =
1127        17, RPC_UNKNOWNADDR = 19, RPC_RPCBFAILURE = 14, RPC_PROGNOTREGISTERED =
1128        15, RPC_N2AXLATEFAILURE = 22, RPC_FAILED = 16, RPC_INTR =
1129        18, RPC_TLIERROR = 20, RPC_UDERROR = 23, RPC_INPROGRESS =
1130        24, RPC_STALERACHANDLE = 25
1131    }
1132     ;
1133    struct rpc_err
1134    {
1135      enum clnt_stat re_status;
1136      union
1137      {
1138        int RE_errno;
1139        enum auth_stat RE_why;
1140        struct
1141        {
1142          u_long low;
1143          u_long high;
1144        }
1145        RE_vers;
1146        struct
1147        {
1148          long s1;
1149          long s2;
1150        }
1151        RE_lb;
1152      }
1153      ru;
1154    }
1155     ;
1156
1157    typedef struct CLIENT
1158    {
1159      struct AUTH *cl_auth;
1160      struct clnt_ops *cl_ops;
1161      caddr_t cl_private;
1162    }
1163    CLIENT;
1164
1165    struct clnt_ops
1166    {
1167      enum clnt_stat (*cl_call) (struct CLIENT *, u_long, xdrproc_t, caddr_t,
1168                                 xdrproc_t, caddr_t, struct timeval);
1169      void (*cl_abort) (void);
1170      void (*cl_geterr) (struct CLIENT *, struct rpc_err *);
1171        bool_t (*cl_freeres) (struct CLIENT *, xdrproc_t, caddr_t);
1172      void (*cl_destroy) (struct CLIENT *);
1173        bool_t (*cl_control) (struct CLIENT *, int, char *);
1174    }
1175     ;
```

1.3.28. rpc/rpc_msg.h

```
1176
1177  enum msg_type
1178  {
1179    CALL, REPLY = 1
1180  }
1181   ;
1182  enum reply_stat
1183  {
1184    MSG_ACCEPTED, MSG_DENIED = 1
1185  }
1186   ;
1187  enum accept_stat
1188  {
1189    SUCCESS, PROG_UNAVAIL = 1, PROG_MISMATCH = 2, PROC_UNAVAIL =
1190      3, GARBAGE_ARGS = 4, SYSTEM_ERR = 5
1191  }
1192   ;
1193  enum reject_stat
1194  {
1195    RPC_MISMATCH, AUTH_ERROR = 1
1196  }
1197   ;
1198
1199  struct accepted_reply
1200  {
1201    struct opaque_auth ar_verf;
1202    enum accept_stat ar_stat;
1203    union
1204    {
1205      struct
1206      {
1207        unsigned long low;
1208        unsigned long high;
1209      }
1210      AR_versions;
1211      struct
1212      {
1213        caddr_t where;
1214        xdrproc_t proc;
1215      }
1216      AR_results;
1217    }
1218    ru;
1219  }
1220   ;
1221
1222  struct rejected_reply
1223  {
1224    enum reject_stat rj_stat;
1225    union
```

```
    {
      struct
      {
        unsigned long low;
        unsigned long high;
      }
      RJ_versions;
      enum auth_stat RJ_why;
    }
    ru;
  }
   ;

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

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

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

1.3.29. rpc/svc.h

```
#define svc_freeargs(xprt,xargs, argsp) (*(xprt)->xp_ops->xp_freeargs)((xprt), (xargs),
(argsp))
#define svc_getargs(xprt,xargs, argsp)  (*(xprt)->xp_ops->xp_getargs)((xprt), (xargs),
(argsp))
#define RPC_ANYSOCK     -1

typedef struct SVCXPRT
{
  int xp_sock;
  u_short xp_port;
  struct xp_ops *xp_ops;
  int xp_addrlen;
  struct sockaddr_in xp_raddr;
  struct opaque_auth xp_verf;
  caddr_t xp_p1;
  caddr_t xp_p2;
  char xp_pad[256];
}
SVCXPRT;

struct svc_req
{
  rpcprog_t rq_prog;
  rpcvers_t rq_vers;
  rpcproc_t rq_proc;
  struct opaque_auth rq_cred;
  caddr_t rq_clntcred;
  SVCXPRT *rq_xprt;
}
 ;

typedef void (*__dispatch_fn_t) (struct svc_req *, SVCXPRT *);

struct xp_ops
{
  bool_t (*xp_recv) (SVCXPRT * __xprt, struct rpc_msg * __msg);
  enum xprt_stat (*xp_stat) (SVCXPRT * __xprt);
    bool_t (*xp_getargs) (SVCXPRT * __xprt, xdrproc_t __xdr_args,
                          caddr_t args_ptr);
    bool_t (*xp_reply) (SVCXPRT * __xprt, struct rpc_msg * __msg);
    bool_t (*xp_freeargs) (SVCXPRT * __xprt, xdrproc_t __xdr_args,
                           caddr_t args_ptr);
  void (*xp_destroy) (SVCXPRT * __xprt);
}
 ;
```

1.3.30. rpc/types.h

```

typedef int bool_t;
typedef int enum_t;
typedef unsigned long rpcprog_t;
```

```
typedef unsigned long rpcvers_t;
typedef unsigned long rpcproc_t;
typedef unsigned long rpcprot_t;
```

1.3.31. rpc/xdr.h

```

enum xdr_op
{
  XDR_ENCODE, XDR_DECODE, XDR_FREE
}
 ;
typedef struct XDR
{
  enum xdr_op x_op;
  struct xdr_ops *x_ops;
  caddr_t x_public;
  caddr_t x_private;
  caddr_t x_base;
  int x_handy;
}
XDR;

struct xdr_ops
{
  bool_t (*x_getlong) (XDR * __xdrs, long *__lp);
  bool_t (*x_putlong) (XDR * __xdrs, long *__lp);
  bool_t (*x_getbytes) (XDR * __xdrs, caddr_t __addr, u_int __len);
  bool_t (*x_putbytes) (XDR * __xdrs, char *__addr, u_int __len);
  u_int (*x_getpostn) (XDR * __xdrs);
  bool_t (*x_setpostn) (XDR * __xdrs, u_int __pos);
  int32_t *(*x_inline) (XDR * __xdrs, int __len);
  void (*x_destroy) (XDR * __xdrs);
    bool_t (*x_getint32) (XDR * __xdrs, int32_t * __ip);
    bool_t (*x_putint32) (XDR * __xdrs, int32_t * __ip);
}
 ;

typedef bool_t (*xdrproc_t) (XDR *, void *, ...);

struct xdr_discrim
{
  int value;
  xdrproc_t proc;
}
 ;
```

1.3.32. sched.h

```

#define SCHED_OTHER     0
#define SCHED_FIFO      1
```

```
#define SCHED_RR        2

struct sched_param
{
  int sched_priority;
}
 ;
```

1.3.33. search.h

```

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

typedef void (*__action_fn_t) (void *__nodep, VISIT __value, int __level);
```

1.3.34. setjmp.h

```

#define setjmp(env)     _setjmp(env)
#define sigsetjmp(a,b)  __sigsetjmp(a,b)

struct __jmp_buf_tag
{
  __jmp_buf __jmpbuf;
  int __mask_was_saved;
  sigset_t __saved_mask;
}
 ;

typedef struct __jmp_buf_tag jmp_buf[1];
typedef jmp_buf sigjmp_buf;
```

1.3.35. signal.h

```

#define SIGRTMAX        (__libc_current_sigrtmax ())
#define SIGRTMIN        (__libc_current_sigrtmin ())
```

```
1413  #define SIG_BLOCK       0
1414  #define SIG_UNBLOCK     1
1415  #define SIG_SETMASK     2
1416  #define NSIG    65
1417
1418  typedef int sig_atomic_t;
1419  struct sigstack
1420  {
1421    void *ss_sp;
1422    int ss_onstack;
1423  }
1424   ;
1425
1426  typedef void (*sighandler_t) (int);
1427  #define SIG_HOLD        ((sighandler_t) 2)
1428  #define SIG_ERR ((sighandler_t)-1)
1429  #define SIG_DFL ((sighandler_t)0)
1430  #define SIG_IGN ((sighandler_t)1)
1431
1432  #define SIGHUP  1
1433  #define SIGUSR1 10
1434  #define SIGSEGV 11
1435  #define SIGUSR2 12
1436  #define SIGPIPE 13
1437  #define SIGALRM 14
1438  #define SIGTERM 15
1439  #define SIGSTKFLT       16
1440  #define SIGCHLD 17
1441  #define SIGCONT 18
1442  #define SIGSTOP 19
1443  #define SIGINT  2
1444  #define SIGTSTP 20
1445  #define SIGTTIN 21
1446  #define SIGTTOU 22
1447  #define SIGURG  23
1448  #define SIGXCPU 24
1449  #define SIGXFSZ 25
1450  #define SIGVTALRM       26
1451  #define SIGPROF 27
1452  #define SIGWINCH        28
1453  #define SIGIO   29
1454  #define SIGQUIT 3
1455  #define SIGPWR  30
1456  #define SIGSYS  31
1457  #define SIGUNUSED       31
1458  #define SIGILL  4
1459  #define SIGTRAP 5
1460  #define SIGABRT 6
1461  #define SIGIOT  6
1462  #define SIGBUS  7
1463  #define SIGFPE  8
1464  #define SIGKILL 9
1465  #define SIGCLD  SIGCHLD
```

```
1466  #define SIGPOLL SIGIO
1467
1468  #define SV_ONSTACK      (1<<0)
1469  #define SV_INTERRUPT    (1<<1)
1470  #define SV_RESETHAND    (1<<2)
1471
1472  typedef union sigval
1473  {
1474    int sival_int;
1475    void *sival_ptr;
1476  }
1477  sigval_t;
1478  #define SIGEV_SIGNAL    0
1479  #define SIGEV_NONE      1
1480  #define SIGEV_THREAD    2
1481
1482  typedef struct sigevent
1483  {
1484    sigval_t sigev_value;
1485    int sigev_signo;
1486    int sigev_notify;
1487    union
1488    {
1489      int _pad[SIGEV_PAD_SIZE];
1490      struct
1491      {
1492        void (*sigev_thread_func) (sigval_t);
1493        void *_attribute;
1494      }
1495      _sigev_thread;
1496    }
1497    _sigev_un;
1498  }
1499  sigevent_t;
1500  #define si_pid  _sifields._kill._pid
1501  #define si_uid  _sifields._kill._uid
1502  #define si_value        _sifields._rt._sigval
1503  #define si_int  _sifields._rt._sigval.sival_int
1504  #define si_ptr  _sifields._rt._sigval.sival_ptr
1505  #define si_status       _sifields._sigchld._status
1506  #define si_stime        _sifields._sigchld._stime
1507  #define si_utime        _sifields._sigchld._utime
1508  #define si_addr _sifields._sigfault._addr
1509  #define si_band _sifields._sigpoll._band
1510  #define si_fd   _sifields._sigpoll._fd
1511  #define si_timer1       _sifields._timer._timer1
1512  #define si_timer2       _sifields._timer._timer2
1513
1514  typedef struct siginfo
1515  {
1516    int si_signo;
1517    int si_errno;
1518    int si_code;
```

```
1519      union
1520      {
1521        int _pad[SI_PAD_SIZE];
1522        struct
1523        {
1524          pid_t _pid;
1525          uid_t _uid;
1526        }
1527        _kill;
1528        struct
1529        {
1530          unsigned int _timer1;
1531          unsigned int _timer2;
1532        }
1533        _timer;
1534        struct
1535        {
1536          pid_t _pid;
1537          uid_t _uid;
1538          sigval_t _sigval;
1539        }
1540        _rt;
1541        struct
1542        {
1543          pid_t _pid;
1544          uid_t _uid;
1545          int _status;
1546          clock_t _utime;
1547          clock_t _stime;
1548        }
1549        _sigchld;
1550        struct
1551        {
1552          void *_addr;
1553        }
1554        _sigfault;
1555        struct
1556        {
1557          int _band;
1558          int _fd;
1559        }
1560        _sigpoll;
1561      }
1562      _sifields;
1563    }
1564    siginfo_t;
1565    #define SI_QUEUE        -1
1566    #define SI_TIMER        -2
1567    #define SI_MESGQ        -3
1568    #define SI_ASYNCIO      -4
1569    #define SI_SIGIO        -5
1570    #define SI_TKILL        -6
1571    #define SI_ASYNCNL      -60
```

```
1572  #define SI_USER 0
1573  #define SI_KERNEL       0x80
1574
1575  #define ILL_ILLOPC      1
1576  #define ILL_ILLOPN      2
1577  #define ILL_ILLADR      3
1578  #define ILL_ILLTRP      4
1579  #define ILL_PRVOPC      5
1580  #define ILL_PRVREG      6
1581  #define ILL_COPROC      7
1582  #define ILL_BADSTK      8
1583
1584  #define FPE_INTDIV      1
1585  #define FPE_INTOVF      2
1586  #define FPE_FLTDIV      3
1587  #define FPE_FLTOVF      4
1588  #define FPE_FLTUND      5
1589  #define FPE_FLTRES      6
1590  #define FPE_FLTINV      7
1591  #define FPE_FLTSUB      8
1592
1593  #define SEGV_MAPERR     1
1594  #define SEGV_ACCERR     2
1595
1596  #define BUS_ADRALN      1
1597  #define BUS_ADRERR      2
1598  #define BUS_OBJERR      3
1599
1600  #define TRAP_BRKPT      1
1601  #define TRAP_TRACE      2
1602
1603  #define CLD_EXITED      1
1604  #define CLD_KILLED      2
1605  #define CLD_DUMPED      3
1606  #define CLD_TRAPPED     4
1607  #define CLD_STOPPED     5
1608  #define CLD_CONTINUED   6
1609
1610  #define POLL_IN 1
1611  #define POLL_OUT        2
1612  #define POLL_MSG        3
1613  #define POLL_ERR        4
1614  #define POLL_PRI        5
1615  #define POLL_HUP        6
1616
1617  typedef struct
1618  {
1619    unsigned long sig[_SIGSET_NWORDS];
1620  }
1621  sigset_t;
1622  #define SA_NOCLDSTOP    0x00000001
1623  #define SA_NOCLDWAIT    0x00000002
1624  #define SA_SIGINFO      0x00000004
```

```
#define SA_ONSTACK      0x08000000
#define SA_RESTART      0x10000000
#define SA_INTERRUPT    0x20000000
#define SA_NODEFER      0x40000000
#define SA_RESETHAND    0x80000000
#define SA_NOMASK       SA_NODEFER
#define SA_ONESHOT      SA_RESETHAND

typedef struct sigaltstack
{
  void *ss_sp;
  int ss_flags;
  size_t ss_size;
}
stack_t;
#define SS_ONSTACK      1
#define SS_DISABLE      2
```

1.3.36. stddef.h

```

#define offsetof(TYPE,MEMBER)   ((size_t)& ((TYPE*)0)->MEMBER)
#define NULL    (0L)

typedef int wchar_t;
```

1.3.37. stdio.h

```

#define EOF     (-1)
#define P_tmpdir        "/tmp"
#define FOPEN_MAX       16
#define L_tmpnam        20
#define FILENAME_MAX    4096
#define BUFSIZ  8192
#define L_ctermid       9
#define L_cuserid       9

typedef struct
{
  off_t __pos;
  mbstate_t __state;
}
fpos_t;
typedef struct
{
  off64_t __pos;
  mbstate_t __state;
}
fpos64_t;

typedef struct _IO_FILE FILE;
```

```
#define _IOFBF  0
#define _IOLBF  1
#define _IONBF  2
```

1.3.38. stdlib.h

```

#define MB_CUR_MAX      (__ctype_get_mb_cur_max())
#define EXIT_SUCCESS    0
#define EXIT_FAILURE    1
#define RAND_MAX        2147483647

typedef int (*__compar_fn_t) (const void *, const void *);
struct random_data
{
  int32_t *fptr;
  int32_t *rptr;
  int32_t *state;
  int rand_type;
  int rand_deg;
  int rand_sep;
  int32_t *end_ptr;
}
 ;

typedef struct
{
  int quot;
  int rem;
}
div_t;

typedef struct
{
  long quot;
  long rem;
}
ldiv_t;

typedef struct
{
  long long quot;
  long long rem;
}
lldiv_t;
```

1.3.39. sys/file.h

```

#define LOCK_SH 1
#define LOCK_EX 2
#define LOCK_NB 4
```

```
#define LOCK_UN 8
```

1.3.40. sys/ipc.h

```

#define IPC_PRIVATE     ((key_t)0)
#define IPC_RMID        0
#define IPC_CREAT       00001000
#define IPC_EXCL        00002000
#define IPC_NOWAIT      00004000
#define IPC_SET 1
#define IPC_STAT        2
```

1.3.41. sys/mman.h

```

#define MAP_FAILED      ((void*)-1)
#define PROT_NONE       0x0
#define MAP_SHARED      0x01
#define MAP_PRIVATE     0x02
#define PROT_READ       0x1
#define MAP_FIXED       0x10
#define PROT_WRITE      0x2
#define MAP_ANONYMOUS   0x20
#define PROT_EXEC       0x4
#define MS_ASYNC        1
#define MS_INVALIDATE   2
#define MS_SYNC 4
#define MAP_ANON        MAP_ANONYMOUS
```

1.3.42. sys/msg.h

```

#define MSG_NOERROR     010000
```

1.3.43. sys/param.h

```

#define NOFILE  256
#define MAXPATHLEN      4096
```

1.3.44. sys/poll.h

```

#define POLLIN  0x0001
#define POLLPRI 0x0002
#define POLLOUT 0x0004
#define POLLERR 0x0008
#define POLLHUP 0x0010
#define POLLNVAL        0x0020

```

```
struct pollfd
{
  int fd;
  short events;
  short revents;
}
 ;
typedef unsigned long nfds_t;
```

1.3.45. sys/resource.h

```

#define RUSAGE_CHILDREN (-1)
#define RUSAGE_BOTH     (-2)
#define RLIM_INFINITY   (~0UL)
#define RLIM_SAVED_CUR  -1
#define RLIM_SAVED_MAX  -1
#define RLIMIT_CPU      0
#define RUSAGE_SELF     0
#define RLIMIT_FSIZE    1
#define RLIMIT_DATA     2
#define RLIMIT_STACK    3
#define RLIMIT_CORE     4
#define RLIMIT_NOFILE   7
#define RLIMIT_AS       9

typedef unsigned long rlim_t;
typedef unsigned long long rlim64_t;
typedef int __rlimit_resource_t;

struct rlimit
{
  rlim_t rlim_cur;
  rlim_t rlim_max;
}
 ;
struct rlimit64
{
  rlim64_t rlim_cur;
  rlim64_t rlim_max;
}
 ;

struct rusage
{
  struct timeval ru_utime;
  struct timeval ru_stime;
  long ru_maxrss;
  long ru_ixrss;
  long ru_idrss;
  long ru_isrss;
  long ru_minflt;
```

```
1802      long ru_majflt;
1803      long ru_nswap;
1804      long ru_inblock;
1805      long ru_oublock;
1806      long ru_msgsnd;
1807      long ru_msgrcv;
1808      long ru_nsignals;
1809      long ru_nvcsw;
1810      long ru_nivcsw;
1811    }
1812     ;
1813
1814    enum __priority_which
1815    {
1816      PRIO_PROCESS, PRIO_PGRP = 1, PRIO_USER = 2
1817    }
1818     ;
1819    #define PRIO_PGRP       PRIO_PGRP
1820    #define PRIO_PROCESS    PRIO_PROCESS
1821    #define PRIO_USER       PRIO_USER
1822
1823    typedef enum __priority_which __priority_which_t;
```

1.3.46. sys/sem.h

```
1824
1825    #define SEM_UNDO        0x1000
1826    #define GETPID  11
1827    #define GETVAL  12
1828    #define GETALL  13
1829    #define GETNCNT 14
1830    #define GETZCNT 15
1831    #define SETVAL  16
1832    #define SETALL  17
1833
1834    struct sembuf
1835    {
1836      short sem_num;
1837      short sem_op;
1838      short sem_flg;
1839    }
1840     ;
```

1.3.47. sys/shm.h

```
1841
1842    #define SHM_RDONLY      010000
1843    #define SHM_W   0200
1844    #define SHM_RND 020000
1845    #define SHM_R   0400
1846    #define SHM_REMAP       040000
1847    #define SHM_LOCK        11
```

```
#define SHM_UNLOCK      12
```

1.3.48. sys/socket.h

```

#define SHUT_RD 0
#define MSG_WAITALL     0x100
#define MSG_TRUNC       0x20
#define MSG_EOR 0x80
#define SIOCGIFCONF     0x8912
#define SIOCGIFFLAGS    0x8913
#define SIOCGIFADDR     0x8915
#define SIOCGIFNETMASK  0x891b
#define MSG_OOB 1
#define SHUT_WR 1
#define MSG_PEEK        2
#define SHUT_RDWR       2
#define MSG_DONTROUTE   4
#define MSG_CTRUNC      8
#define PF_UNSPEC       AF_UNSPEC

struct linger
{
  int l_onoff;
  int l_linger;
}
 ;
struct cmsghdr
{
  size_t cmsg_len;
  int cmsg_level;
  int cmsg_type;
}
 ;
struct iovec
{
  void *iov_base;
  size_t iov_len;
}
 ;

typedef unsigned short sa_family_t;
typedef unsigned int socklen_t;

struct sockaddr
{
  sa_family_t sa_family;
  char sa_data[14];
}
 ;
struct sockaddr_storage
{
```

```
    sa_family_t ss_family;
    __ss_aligntype __ss_align;
    char __ss_padding[(128 - (2 * sizeof (__ss_aligntype)))];
  }
   ;

  struct msghdr
  {
    void *msg_name;
    int msg_namelen;
    struct iovec *msg_iov;
    size_t msg_iovlen;
    void *msg_control;
    size_t msg_controllen;
    unsigned int msg_flags;
  }
   ;
  #define AF_UNSPEC       0
  #define AF_UNIX 1
  #define AF_INET6        10
  #define AF_INET 2

  #define PF_INET AF_INET
  #define PF_INET6        AF_INET6
  #define PF_UNIX AF_UNIX

  #define SOCK_STREAM     1
  #define SOCK_PACKET     10
  #define SOCK_DGRAM      2
  #define SOCK_RAW        3
  #define SOCK_RDM        4
  #define SOCK_SEQPACKET  5

  #define SOL_SOCKET      1
  #define SO_DEBUG        1
  #define SO_OOBINLINE    10
  #define SO_NO_CHECK     11
  #define SO_PRIORITY     12
  #define SO_LINGER       13
  #define SO_REUSEADDR    2
  #define SOL_RAW 255
  #define SO_TYPE 3
  #define SO_ERROR        4
  #define SO_DONTROUTE    5
  #define SO_BROADCAST    6
  #define SO_SNDBUF       7
  #define SO_RCVBUF       8
  #define SO_KEEPALIVE    9
```

1.3.49. sys/stat.h

```
#define S_ISBLK(m)      (((m)& S_IFMT)==S_IFBLK)
#define S_ISCHR(m)      (((m)& S_IFMT)==S_IFCHR)
#define S_ISDIR(m)      (((m)& S_IFMT)==S_IFDIR)
#define S_ISFIFO(m)     (((m)& S_IFMT)==S_IFIFO)
#define S_ISLNK(m)      (((m)& S_IFMT)==S_IFLNK)
#define S_ISREG(m)      (((m)& S_IFMT)==S_IFREG)
#define S_ISSOCK(m)     (((m)& S_IFMT)==S_IFSOCK)
#define S_TYPEISMQ(buf) ((buf)->st_mode - (buf)->st_mode)
#define S_TYPEISSEM(buf)        ((buf)->st_mode - (buf)->st_mode)
#define S_TYPEISSHM(buf)        ((buf)->st_mode - (buf)->st_mode)
#define S_IRWXU (S_IREAD|S_IWRITE|S_IEXEC)
#define S_IROTH (S_IRGRP>>3)
#define S_IRGRP (S_IRUSR>>3)
#define S_IRWXO (S_IRWXG>>3)
#define S_IRWXG (S_IRWXU>>3)
#define S_IWOTH (S_IWGRP>>3)
#define S_IWGRP (S_IWUSR>>3)
#define S_IXOTH (S_IXGRP>>3)
#define S_IXGRP (S_IXUSR>>3)
#define S_ISVTX 01000
#define S_IXUSR 0x0040
#define S_IWUSR 0x0080
#define S_IRUSR 0x0100
#define S_ISGID 0x0400
#define S_ISUID 0x0800
#define S_IFIFO 0x1000
#define S_IFCHR 0x2000
#define S_IFDIR 0x4000
#define S_IFBLK 0x6000
#define S_IFREG 0x8000
#define S_IFLNK 0xa000
#define S_IFSOCK        0xc000
#define S_IFMT  0xf000
#define st_atime        st_atim.tv_sec
#define st_ctime        st_ctim.tv_sec
#define st_mtime        st_mtim.tv_sec
#define S_IREAD S_IRUSR
#define S_IWRITE        S_IWUSR
#define S_IEXEC S_IXUSR
```

1.3.50. sys/time.h

```

#define ITIMER_REAL     0
#define ITIMER_VIRTUAL  1
#define ITIMER_PROF     2

struct timezone
{
  int tz_minuteswest;
  int tz_dsttime;
}
```

```
 ;

typedef int __itimer_which_t;

struct timespec
{
  time_t tv_sec;
  long tv_nsec;
}
 ;

struct timeval
{
  time_t tv_sec;
  suseconds_t tv_usec;
}
 ;

struct itimerval
{
  struct timeval it_interval;
  struct timeval it_value;
}
 ;
```

1.3.51. sys/timeb.h

```

struct timeb
{
  time_t time;
  unsigned short millitm;
  short timezone;
  short dstflag;
}
 ;
```

1.3.52. sys/times.h

```

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

1.3.53. sys/types.h

```
```

```
2038  #define FD_ISSET(d,set) ((set)->fds_bits[((d)/(8*sizeof(long)))]&
2039  (1<<((d)%(8*sizeof(long)))))
2040  #define FD_CLR(d,set)   ((set)->fds_bits[((d)/(8*sizeof(long)))]&
2041  =~(1<<((d)%(8*sizeof(long)))))
2042  #define FD_SET(d,set)
2043  ((set)->fds_bits[((d)/(8*sizeof(long)))]|=(1<<((d)%(8*sizeof(long)))))
2044  #define FALSE   0
2045  #define TRUE    1
2046  #define FD_SETSIZE      1024
2047  #define FD_ZERO(fdsetp) bzero(fdsetp, sizeof(*(fdsetp)))
2048
2049  typedef signed char int8_t;
2050  typedef short int16_t;
2051  typedef int int32_t;
2052  typedef unsigned char u_int8_t;
2053  typedef unsigned short u_int16_t;
2054  typedef unsigned int u_int32_t;
2055  typedef unsigned int uid_t;
2056  typedef int pid_t;
2057  typedef unsigned long off_t;
2058  typedef int key_t;
2059  typedef long suseconds_t;
2060  typedef unsigned int u_int;
2061  typedef struct
2062  {
2063    int __val[2];
2064  }
2065  fsid_t;
2066  typedef unsigned int useconds_t;
2067  typedef unsigned long blksize_t;
2068  typedef long fd_mask;
2069  typedef int timer_t;
2070  typedef int clockid_t;
2071
2072  typedef unsigned int id_t;
2073
2074  typedef unsigned long long ino64_t;
2075  typedef long long loff_t;
2076  typedef unsigned long blkcnt_t;
2077  typedef unsigned long fsblkcnt_t;
2078  typedef unsigned long fsfilcnt_t;
2079  typedef unsigned long long blkcnt64_t;
2080  typedef unsigned long long fsblkcnt64_t;
2081  typedef unsigned long long fsfilcnt64_t;
2082  typedef unsigned char u_char;
2083  typedef unsigned short u_short;
2084  typedef unsigned long u_long;
2085
2086  typedef unsigned long ino_t;
2087  typedef unsigned int gid_t;
2088  typedef unsigned long long dev_t;
2089  typedef unsigned int mode_t;
2090  typedef unsigned long nlink_t;
```

```
typedef char *caddr_t;

typedef struct
{
  unsigned long fds_bits[__FDSET_LONGS];
}
fd_set;

typedef long clock_t;
typedef long time_t;
```

1.3.54. sys/un.h

```

#define UNIX_PATH_MAX   108

struct sockaddr_un
{
  sa_family_t sun_family;
  char sun_path[UNIX_PATH_MAX];
}
 ;
```

1.3.55. sys/utsname.h

```

#define SYS_NMLN        65

struct utsname
{
  char sysname[65];
  char nodename[65];
  char release[65];
  char version[65];
  char machine[65];
  char domainname[65];
}
 ;
```

1.3.56. sys/wait.h

```

#define WIFSIGNALED(status)     (!WIFSTOPPED(status) & & !WIFEXITED(status))
#define WIFSTOPPED(status)      (((status) &  0xff) == 0x7f)
#define WEXITSTATUS(status)     (((status) &  0xff00) >> 8)
#define WTERMSIG(status)        ((status) &  0x7f)
#define WCOREDUMP(status)       ((status) &  0x80)
#define WIFEXITED(status)       (WTERMSIG(status) == 0)
#define WNOHANG 0x00000001
#define WUNTRACED       0x00000002
#define WCOREFLAG       0x80
#define WSTOPSIG(status)        WEXITSTATUS(status)
```

```
2134
2135  typedef enum
2136  {
2137    P_ALL, P_PID, P_PGID
2138  }
2139  idtype_t;
```

1.3.57. syslog.h

```
2140
2141  #define LOG_EMERG       0
2142  #define LOG_PRIMASK     0x07
2143  #define LOG_ALERT       1
2144  #define LOG_CRIT        2
2145  #define LOG_ERR 3
2146  #define LOG_WARNING     4
2147  #define LOG_NOTICE      5
2148  #define LOG_INFO        6
2149  #define LOG_DEBUG       7
2150
2151  #define LOG_KERN        (0<<3)
2152  #define LOG_AUTHPRIV    (10<<3)
2153  #define LOG_FTP (11<<3)
2154  #define LOG_USER        (1<<3)
2155  #define LOG_MAIL        (2<<3)
2156  #define LOG_DAEMON      (3<<3)
2157  #define LOG_AUTH        (4<<3)
2158  #define LOG_SYSLOG      (5<<3)
2159  #define LOG_LPR (6<<3)
2160  #define LOG_NEWS        (7<<3)
2161  #define LOG_UUCP        (8<<3)
2162  #define LOG_CRON        (9<<3)
2163  #define LOG_FACMASK     0x03f8
2164
2165  #define LOG_LOCAL0      (16<<3)
2166  #define LOG_LOCAL1      (17<<3)
2167  #define LOG_LOCAL2      (18<<3)
2168  #define LOG_LOCAL3      (19<<3)
2169  #define LOG_LOCAL4      (20<<3)
2170  #define LOG_LOCAL5      (21<<3)
2171  #define LOG_LOCAL6      (22<<3)
2172  #define LOG_LOCAL7      (23<<3)
2173
2174  #define LOG_UPTO(pri)   ((1 << ((pri)+1)) - 1)
2175  #define LOG_MASK(pri)   (1 << (pri))
2176
2177  #define LOG_PID 0x01
2178  #define LOG_CONS        0x02
2179  #define LOG_ODELAY      0x04
2180  #define LOG_NDELAY      0x08
2181  #define LOG_NOWAIT      0x10
2182  #define LOG_PERROR      0x20
```

1.3.58. termios.h

```
2183
2184  #define TCIFLUSH        0
2185  #define TCOOFF  0
2186  #define TCSANOW 0
2187  #define BS0     0000000
2188  #define CR0     0000000
2189  #define FF0     0000000
2190  #define NL0     0000000
2191  #define TAB0    0000000
2192  #define VT0     0000000
2193  #define OPOST   0000001
2194  #define OCRNL   0000010
2195  #define ONOCR   0000020
2196  #define ONLRET  0000040
2197  #define OFILL   0000100
2198  #define OFDEL   0000200
2199  #define NL1     0000400
2200  #define TCOFLUSH        1
2201  #define TCOON   1
2202  #define TCSADRAIN       1
2203  #define TCIOFF  2
2204  #define TCIOFLUSH       2
2205  #define TCSAFLUSH       2
2206  #define TCION   3
2207
2208  typedef unsigned int speed_t;
2209  typedef unsigned char cc_t;
2210  typedef unsigned int tcflag_t;
2211  #define NCCS    32
2212
2213  struct termios
2214  {
2215    tcflag_t c_iflag;
2216    tcflag_t c_oflag;
2217    tcflag_t c_cflag;
2218    tcflag_t c_lflag;
2219    cc_t c_line;
2220    cc_t c_cc[NCCS];
2221    speed_t c_ispeed;
2222    speed_t c_ospeed;
2223  }
2224   ;
2225  #define VINTR   0
2226  #define VQUIT   1
2227  #define VLNEXT  15
2228  #define VERASE  2
2229  #define VKILL   3
2230  #define VEOF    4
2231
2232  #define IGNBRK  0000001
```

```
#define BRKINT  0000002
#define IGNPAR  0000004
#define PARMRK  0000010
#define INPCK   0000020
#define ISTRIP  0000040
#define INLCR   0000100
#define IGNCR   0000200
#define ICRNL   0000400
#define IXANY   0004000
#define IMAXBEL 0020000

#define CS5     0000000

#define ECHO    0000010

#define B0      0000000
#define B50     0000001
#define B75     0000002
#define B110    0000003
#define B134    0000004
#define B150    0000005
#define B200    0000006
#define B300    0000007
#define B600    0000010
#define B1200   0000011
#define B1800   0000012
#define B2400   0000013
#define B4800   0000014
#define B9600   0000015
#define B19200  0000016
#define B38400  0000017
```

1.3.59. time.h

```

#define CLK_TCK ((clock_t)__sysconf(2))
#define CLOCK_REALTIME  0
#define TIMER_ABSTIME   1
#define CLOCKS_PER_SEC  1000000l

struct tm
{
  int tm_sec;
  int tm_min;
  int tm_hour;
  int tm_mday;
  int tm_mon;
  int tm_year;
  int tm_wday;
  int tm_yday;
  int tm_isdst;
  long tm_gmtoff;
```

```
  char *tm_zone;
}
 ;
struct itimerspec
{
  struct timespec it_interval;
  struct timespec it_value;
}
 ;
```

1.3.60. ulimit.h

```

#define UL_GETFSIZE     1
#define UL_SETFSIZE     2
```

1.3.61. unistd.h

```

#define SEEK_SET        0
#define STDIN_FILENO    0
#define SEEK_CUR        1
#define STDOUT_FILENO   1
#define SEEK_END        2
#define STDERR_FILENO   2

typedef long long off64_t;
#define F_OK    0
#define X_OK    1
#define W_OK    2
#define R_OK    4

#define _POSIX_VDISABLE '\0'
#define _POSIX_CHOWN_RESTRICTED 1
#define _POSIX_JOB_CONTROL      1
#define _POSIX_NO_TRUNC 1
#define _POSIX_SHELL    1
#define _POSIX_FSYNC    200112
#define _POSIX_MAPPED_FILES     200112
#define _POSIX_MEMLOCK  200112
#define _POSIX_MEMLOCK_RANGE    200112
#define _POSIX_MEMORY_PROTECTION        200112
#define _POSIX_SEMAPHORES       200112
#define _POSIX_SHARED_MEMORY_OBJECTS    200112
#define _POSIX_TIMERS   200112
#define _POSIX2_C_BIND  200112L
#define _POSIX2_VERSION 200112L
#define _POSIX_THREADS  200112L
#define _POSIX_VERSION  200112L

#define _PC_LINK_MAX    0
#define _PC_MAX_CANON   1
```

```
2328    #define _PC_ASYNC_IO    10
2329    #define _PC_PRIO_IO     11
2330    #define _PC_FILESIZEBITS        13
2331    #define _PC_REC_INCR_XFER_SIZE  14
2332    #define _PC_REC_MIN_XFER_SIZE   16
2333    #define _PC_REC_XFER_ALIGN      17
2334    #define _PC_ALLOC_SIZE_MIN      18
2335    #define _PC_MAX_INPUT   2
2336    #define _PC_2_SYMLINKS  20
2337    #define _PC_NAME_MAX    3
2338    #define _PC_PATH_MAX    4
2339    #define _PC_PIPE_BUF    5
2340    #define _PC_CHOWN_RESTRICTED    6
2341    #define _PC_NO_TRUNC    7
2342    #define _PC_VDISABLE    8
2343    #define _PC_SYNC_IO     9
2344
2345    #define _SC_ARG_MAX     0
2346    #define _SC_CHILD_MAX   1
2347    #define _SC_PRIORITY_SCHEDULING 10
2348    #define _SC_TIMERS      11
2349    #define _SC_ASYNCHRONOUS_IO     12
2350    #define _SC_XBS5_ILP32_OFF32    125
2351    #define _SC_XBS5_ILP32_OFFBIG   126
2352    #define _SC_XBS5_LP64_OFF64     127
2353    #define _SC_XBS5_LPBIG_OFFBIG   128
2354    #define _SC_XOPEN_LEGACY        129
2355    #define _SC_PRIORITIZED_IO      13
2356    #define _SC_XOPEN_REALTIME      130
2357    #define _SC_XOPEN_REALTIME_THREADS      131
2358    #define _SC_ADVISORY_INFO       132
2359    #define _SC_BARRIERS    133
2360    #define _SC_CLOCK_SELECTION     137
2361    #define _SC_CPUTIME     138
2362    #define _SC_THREAD_CPUTIME      139
2363    #define _SC_SYNCHRONIZED_IO     14
2364    #define _SC_MONOTONIC_CLOCK     149
2365    #define _SC_FSYNC       15
2366    #define _SC_READER_WRITER_LOCKS 153
2367    #define _SC_SPIN_LOCKS  154
2368    #define _SC_REGEXP      155
2369    #define _SC_SHELL       157
2370    #define _SC_SPAWN       159
2371    #define _SC_MAPPED_FILES        16
2372    #define _SC_SPORADIC_SERVER     160
2373    #define _SC_THREAD_SPORADIC_SERVER      161
2374    #define _SC_TIMEOUTS    164
2375    #define _SC_TYPED_MEMORY_OBJECTS        165
2376    #define _SC_2_PBS_ACCOUNTING    169
2377    #define _SC_MEMLOCK     17
2378    #define _SC_2_PBS_LOCATE        170
2379    #define _SC_2_PBS_MESSAGE       171
2380    #define _SC_2_PBS_TRACK 172
```

```
2381    #define _SC_SYMLOOP_MAX 173
2382    #define _SC_2_PBS_CHECKPOINT    175
2383    #define _SC_V6_ILP32_OFF32      176
2384    #define _SC_V6_ILP32_OFFBIG     177
2385    #define _SC_V6_LP64_OFF64       178
2386    #define _SC_V6_LPBIG_OFFBIG     179
2387    #define _SC_MEMLOCK_RANGE       18
2388    #define _SC_HOST_NAME_MAX       180
2389    #define _SC_TRACE       181
2390    #define _SC_TRACE_EVENT_FILTER  182
2391    #define _SC_TRACE_INHERIT       183
2392    #define _SC_TRACE_LOG   184
2393    #define _SC_MEMORY_PROTECTION   19
2394    #define _SC_CLK_TCK     2
2395    #define _SC_MESSAGE_PASSING     20
2396    #define _SC_SEMAPHORES  21
2397    #define _SC_SHARED_MEMORY_OBJECTS       22
2398    #define _SC_AIO_LISTIO_MAX      23
2399    #define _SC_AIO_MAX     24
2400    #define _SC_AIO_PRIO_DELTA_MAX  25
2401    #define _SC_DELAYTIMER_MAX      26
2402    #define _SC_MQ_OPEN_MAX 27
2403    #define _SC_MQ_PRIO_MAX 28
2404    #define _SC_VERSION     29
2405    #define _SC_NGROUPS_MAX 3
2406    #define _SC_PAGESIZE    30
2407    #define _SC_PAGE_SIZE   30
2408    #define _SC_RTSIG_MAX   31
2409    #define _SC_SEM_NSEMS_MAX       32
2410    #define _SC_SEM_VALUE_MAX       33
2411    #define _SC_SIGQUEUE_MAX        34
2412    #define _SC_TIMER_MAX   35
2413    #define _SC_BC_BASE_MAX 36
2414    #define _SC_BC_DIM_MAX  37
2415    #define _SC_BC_SCALE_MAX        38
2416    #define _SC_BC_STRING_MAX       39
2417    #define _SC_OPEN_MAX    4
2418    #define _SC_COLL_WEIGHTS_MAX    40
2419    #define _SC_EXPR_NEST_MAX       42
2420    #define _SC_LINE_MAX    43
2421    #define _SC_RE_DUP_MAX  44
2422    #define _SC_2_VERSION   46
2423    #define _SC_2_C_BIND    47
2424    #define _SC_2_C_DEV     48
2425    #define _SC_2_FORT_DEV  49
2426    #define _SC_STREAM_MAX  5
2427    #define _SC_2_FORT_RUN  50
2428    #define _SC_2_SW_DEV    51
2429    #define _SC_2_LOCALEDEF 52
2430    #define _SC_TZNAME_MAX  6
2431    #define _SC_IOV_MAX     60
2432    #define _SC_THREADS     67
2433    #define _SC_THREAD_SAFE_FUNCTIONS       68
```

```
2434    #define _SC_GETGR_R_SIZE_MAX    69
2435    #define _SC_JOB_CONTROL 7
2436    #define _SC_GETPW_R_SIZE_MAX    70
2437    #define _SC_LOGIN_NAME_MAX      71
2438    #define _SC_TTY_NAME_MAX        72
2439    #define _SC_THREAD_DESTRUCTOR_ITERATIONS        73
2440    #define _SC_THREAD_KEYS_MAX     74
2441    #define _SC_THREAD_STACK_MIN    75
2442    #define _SC_THREAD_THREADS_MAX  76
2443    #define _SC_THREAD_ATTR_STACKADDR       77
2444    #define _SC_THREAD_ATTR_STACKSIZE       78
2445    #define _SC_THREAD_PRIORITY_SCHEDULING  79
2446    #define _SC_SAVED_IDS   8
2447    #define _SC_THREAD_PRIO_INHERIT 80
2448    #define _SC_THREAD_PRIO_PROTECT 81
2449    #define _SC_THREAD_PROCESS_SHARED       82
2450    #define _SC_ATEXIT_MAX  87
2451    #define _SC_PASS_MAX    88
2452    #define _SC_XOPEN_VERSION       89
2453    #define _SC_REALTIME_SIGNALS    9
2454    #define _SC_XOPEN_UNIX  91
2455    #define _SC_XOPEN_CRYPT 92
2456    #define _SC_XOPEN_ENH_I18N      93
2457    #define _SC_XOPEN_SHM   94
2458    #define _SC_2_CHAR_TERM 95
2459    #define _SC_2_C_VERSION 96
2460    #define _SC_2_UPE       97
2461
2462    #define _CS_PATH        0
2463    #define _POSIX_REGEXP   1
2464    #define _CS_XBS5_ILP32_OFF32_CFLAGS     1100
2465    #define _CS_XBS5_ILP32_OFF32_LDFLAGS    1101
2466    #define _CS_XBS5_ILP32_OFF32_LIBS       1102
2467    #define _CS_XBS5_ILP32_OFF32_LINTFLAGS  1103
2468    #define _CS_XBS5_ILP32_OFFBIG_CFLAGS    1104
2469    #define _CS_XBS5_ILP32_OFFBIG_LDFLAGS   1105
2470    #define _CS_XBS5_ILP32_OFFBIG_LIBS      1106
2471    #define _CS_XBS5_ILP32_OFFBIG_LINTFLAGS 1107
2472    #define _CS_XBS5_LP64_OFF64_CFLAGS      1108
2473    #define _CS_XBS5_LP64_OFF64_LDFLAGS     1109
2474    #define _CS_XBS5_LP64_OFF64_LIBS        1110
2475    #define _CS_XBS5_LP64_OFF64_LINTFLAGS   1111
2476    #define _CS_XBS5_LPBIG_OFFBIG_CFLAGS    1112
2477    #define _CS_XBS5_LPBIG_OFFBIG_LDFLAGS   1113
2478    #define _CS_XBS5_LPBIG_OFFBIG_LIBS      1114
2479    #define _CS_XBS5_LPBIG_OFFBIG_LINTFLAGS 1115
2480
2481    #define _XOPEN_REALTIME 1
2482    #define _XOPEN_XPG4     1
2483    #define _XOPEN_XCU_VERSION      4
2484    #define _XOPEN_VERSION  500
2485
2486    #define F_ULOCK 0
```

```
#define F_LOCK  1
#define F_TLOCK 2
#define F_TEST  3
```

1.3.62. utime.h

```

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

1.3.63. utmp.h

```

#define UT_HOSTSIZE     256
#define UT_LINESIZE     32
#define UT_NAMESIZE     32

struct exit_status
{
  short e_termination;
  short e_exit;
}
 ;

#define EMPTY   0
#define RUN_LVL 1
#define BOOT_TIME       2
#define NEW_TIME        3
#define OLD_TIME        4
#define INIT_PROCESS    5
#define LOGIN_PROCESS   6
#define USER_PROCESS    7
#define DEAD_PROCESS    8
#define ACCOUNTING      9
```

1.3.64. wchar.h

```

#define WEOF    (0xffffffffu)
#define WCHAR_MAX       0x7FFFFFFF
#define WCHAR_MIN       0x80000000
```

1.3.65. wctype.h

```

typedef unsigned long wctype_t;
typedef unsigned int wint_t;
```

```
typedef const int32_t *wctrans_t;
typedef struct
{
  int count;
  wint_t value;
}
__mbstate_t;

typedef __mbstate_t mbstate_t;
```

1.3.66. wordexp.h

```

enum
{
  WRDE_DOOFFS, WRDE_APPEND, WRDE_NOCMD, WRDE_REUSE, WRDE_SHOWERR, WRDE_UNDEF,
    __WRDE_FLAGS
}
 ;

typedef struct
{
  int we_wordc;
  char **we_wordv;
  int we_offs;
}
wordexp_t;

enum
{
  WRDE_NOSYS, WRDE_NOSPACE, WRDE_BADCHAR, WRDE_BADVAL, WRDE_CMDSUB,
    WRDE_SYNTAX
}
 ;
```

1.4. Interface Definitions for libc

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

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

_IO_feof

Name

_IO_feof — alias for feof

Synopsis

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

Description

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

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

_IO_getc

Name

_IO_getc — alias for getc

Synopsis

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

Description

_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 or error.

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

_IO_putc

Name

_IO_putc — alias for putc

Synopsis

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

Description

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

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

_IO_puts

Name

_IO_puts — alias for puts

Synopsis

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

Description

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

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

__assert_fail

Name

__assert_fail — abort the program after false assertion

Synopsis

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

Description

The __assert_fail function is used to implement the assert interface of ISO POSIX (2003). The __assert_fail function shall print the given *file* filename, *line* line number, *function* function name and a message on the standard error stream in an unspecified format, and abort program execution via the abort function. For example:

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

If *function* is NULL, __assert_fail shall omit information about the function.

assertion, *file*, and *line* shall be non-NULL.

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

__ctype_b_loc

Name

__ctype_b_loc — accessor function for __ctype_b array for ctype functions

Synopsis

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

Description

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

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

Return Value

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

__ctype_get_mb_cur_max

Name

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

Synopsis

```
size_t __ctype_get_mb_cur_max(void);
```

Description

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

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

__ctype_tolower_loc

Name

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

Synopsis

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

Description

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

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

__ctype_toupper_loc

Name

__ctype_toupper_loc — accessor function for __ctype_b_toupper array for ctype toupper() function

Synopsis

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

Description

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

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

__cxa_atexit

Name

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

Synopsis

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

Description

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

The __cxa_atexit function is used to implement atexit, as described in ISO POSIX (2003). Calling

```
atexit(func)
```

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

```
__cxa_atexit(func, NULL, NULL)
```

.

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

__daylight

Name

__daylight — Daylight savings time flag

Synopsis

```
int __daylight;
```

Description

The integer variable __daylight shall implement the daylight savings time flag daylight as specified in the ISO POSIX (2003) header file <time.h>.

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

__environ

Name

__environ — alias for environ - user environment

Synopsis

```
extern char **__environ;
```

Description

__environ is an alias for environ - user environment.

__environ has the same specification as environ.

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

__errno_location

Name

__errno_location — address of errno variable

Synopsis

```
int *__errno_location(void);
```

Description

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

__fpending

Name

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

Synopsis

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

Description

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

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

__getpagesize

Name

__getpagesize — alias for getpagesize - get current page size

Synopsis

```
int __getpagesize(void);
```

Description

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

__getpagesize has the same specification as getpagesize.

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

__getpgid

Name

__getpgid — get the process group id

Synopsis

```
pid_t __getpgid(pid_t pid);
```

Description

__getpgid has the same specification as getpgid.

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

__h_errno_location

Name

__h_errno_location — address of h_errno variable

Synopsis

```
int *__h_errno_location(void);
```

Description

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

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

__isinf

Name

__isinf — test for infinity

Synopsis

```
int __isinf(double arg);
```

Description

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

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

__isinff

Name

__isinff — test for infinity

Synopsis

```
int __isinff(float arg);
```

Description

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

__isinff is not in the source standard; it is only in the binary standard.

__isinfl

Name

__isinfl — test for infinity

Synopsis

```
int __isinfl(long double arg);
```

Description

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

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

__isnan

Name

__isnan — test for infinity

Synopsis

```
int __isnan(double arg);
```

Description

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

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

__isnanf

Name

__isnanf — test for infinity

Synopsis

```
int __isnanf(float arg);
```

Description

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

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

__isnanl

Name

__isnanl — test for infinity

Synopsis

```
int __isnanl(long double arg);
```

Description

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

__isnanl is not in the source standard; it is only in the binary standard.

__libc_current_sigrtmax

Name

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

Synopsis

```
int __libc_current_sigrtmax(void);
```

Description

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

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

__libc_current_sigrtmin

Name

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

Synopsis

```
int __libc_current_sigrtmin(void);
```

Description

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

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

__libc_start_main

Name

__libc_start_main — initialization routine

Synopsis

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

Description

The __libc_start_main function shall initialize the process, call the *main* function with appropriate arguments, and handle the return from main.

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

__lxstat

Name

__lxstat — inline wrapper around call to lxstat

Synopsis

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

Description

__lxstat is an inline wrapper around call to lxstat.

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

__mempcpy

Name

__mempcpy — copy given number of bytes of source to destination

Synopsis

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

Description

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

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

__rawmemchr

Name

__rawmemchr — scan memory

Synopsis

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

Description

__rawmemchr searches in *s* for *c*.

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

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

__register_atfork

Name

__register_atfork — alias for register_atfork

Synopsis

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

Description

__register_atfork implements pthread_atfork as specified in ISO POSIX (2003). The additional parameter *__dso_handle* allows a shared object to pass in it's handle so that functions registered by __register_atfork can be unregistered by the runtime when the shared object is unloaded.

__sigsetjmp

Name

__sigsetjmp — save stack context for non-local goto

Synopsis

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

Description

__sigsetjmp has the same behavior as sigsetjmp as specified by ISO POSIX (2003).

__sigsetjmp is not in the source standard; it is only in the binary standard.

__stpcpy

Name

__stpcpy — copy a string returning a pointer to its end

Synopsis

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

Description

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

Return Value

__stpcpy returns a pointer to the end of the string *dest* (that is, the address of the terminating NULL character) rather than the beginning.

__stpcpy has the same specification as stpcpy.

__stpcpy is not in the source standard; it is only in the binary standard.

__strdup

Name

__strdup — alias for strdup

Synopsis

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

Description

__strdup has the same specification as strdup.

__strdup is not in the source standard; it is only in the binary standard.

__strtod_internal

Name

__strtod_internal — underlying function for strtod

Synopsis

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

Description

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

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

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

__strtof_internal

Name

__strtof_internal — underlying function for strtof

Synopsis

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

Description

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

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

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

__strtok_r

Name

__strtok_r — alias for strtok_r

Synopsis

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

Description

__strtok_r has the same specification as strtok_r.

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

__strtol_internal

Name

__strtol_internal — alias for strtol

Synopsis

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

Description

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

__strtol_internal(*__nptr*, *__endptr*, *__base*, 0) has the same specification as strtol(*__nptr*, *__endptr*, *__base*).

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

__strtold_internal

Name

__strtold_internal — underlying function for strtold

Synopsis

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

Description

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

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

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

__strtoll_internal

Name

__strtoll_internal — underlying function for strtoll

Synopsis

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

Description

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

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

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

__strtoul_internal

Name

__strtoul_internal — underlying function for strtoul

Synopsis

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

Description

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

__strtoul_internal(*__nptr*, *__endptr*, *__base*, 0) has the same specification as strtoul(*__nptr*, *__endptr*, *__base*).

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

__strtoull_internal

Name

__strtoull_internal — underlying function for strtoull

Synopsis

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

Description

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

__strtoull_internal(*__nptr*, *__endptr*, *__base*, 0) has the same specification as strtoull(*__nptr*, *__endptr*, *__base*).

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

__sysconf

Name

__sysconf — get configuration information at runtime

Synopsis

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

Description

__sysconf gets configuration information at runtime.

__sysconf is weak alias to sysconf.

__sysconf has the same specification as sysconf.

__sysconf is not in the source standard; it is only in the binary standard.

__sysv_signal

Name

__sysv_signal — signal handling

Synopsis

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

Description

__sysv_signal has the same behavior as signal as specified by ISO POSIX (2003).

__sysv_signal is not in the source standard; it is only in the binary standard.

__timezone

Name

— global variable containing timezone

Synopsis

```
long int __timezone;
```

Description

`__timezone` has the same specification as `timezone` in the *ISO POSIX (2003)*

__tzname

Name

— global variable containing the timezone

Synopsis

```
char *__tzname[2];
```

Description

`__tzname` has the same specification as `tzname` in the *ISO POSIX (2003)*.

Note that the array size of `2` is explicit in the *ISO POSIX (2003)*, but not in the *SUSv2*.

__wcstod_internal

Name

`__wcstod_internal` — underlying function for wcstod

Synopsis

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

Description

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

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

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

__wcstof_internal

Name

__wcstof_internal — underlying function for wcstof

Synopsis

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

Description

group shall be 0 or the behavior of __wcstof_internal is undefined.

__wcstof_internal(*nptr*, *endptr*, 0) has the same specification as wcstof(*nptr*, *endptr*).

__wcstof_internal is not in the source standard; it is only in the binary standard.

__wcstol_internal

Name

__wcstol_internal — underlying function for wcstol

Synopsis

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

Description

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

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

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

__wcstold_internal

Name

__wcstold_internal — underlying function for wcstold

Synopsis

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

Description

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

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

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

__wcstoul_internal

Name

__wcstoul_internal — underlying function for wcstoul

Synopsis

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

Description

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

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

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

__xmknod

Name

__xmknod — make block or character special file

Synopsis

```
int __xmknod(int ver, const char *path, mode_t mode, dev_t *dev);
```

Description

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

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

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

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

__xstat

Name

__xstat — Get File Status

Synopsis

```
#include <sys/stat.h>
#include <unistd.h>
int __xstat(int ver, const char *path, (struct stat *stat_buf));
int __lxstat(int ver, const char *path, (struct stat *stat_buf));
int __fxstat(int ver, int fildes, (struct stat *stat_buf));
```

Description

The functions __xstat, __lxstat, and __fxstat shall implement the ISO POSIX (2003) functions stat, lstat, and fstat respectively.

ver shall be 3 or the behavior of these functions is undefined.

__xstat(3, *path*, *stat_buf*) shall behave as stat(*path*, *stat_buf*) as specified by ISO POSIX (2003).

__lxstat(3, *path*, *stat_buf*) shall behave as lstat(*path*, *stat_buf*) as specified by ISO POSIX (2003).

__fxstat(3, *fildes*, *stat_buf*) shall behave as fstat(*fildes*, *stat_buf*) as specified by ISO POSIX (2003).

__xstat, __lxstat, and __fxstat are not in the source standard; they are only in the binary standard.

stat, lstat, and fstat are not in the binary standard; they are only in the source standard.

__xstat64

Name

__xstat64 — Get File Status

Synopsis

```
#define _LARGEFILE_SOURCE 1
#include <sys/stat.h>
#include <unistd.h>
int __xstat64(int ver, const char *path, (struct stat64 *stat_buf));
int __lxstat64(int ver, const char *path, (struct stat64 *stat_buf));
int __fxstat64(int ver, int fildes, (struct stat64 *stat_buf));
```

Description

The functions __xstat64, __lxstat64, and __fxstat64 shall implement the Large File Support functions stat64, lstat64, and fstat64 respectively.

ver shall be 3 or the behavior of these functions is undefined.

__xstat64(3, *path*, *stat_buf*) shall behave as stat(*path*, *stat_buf*) as specified by Large File Support.

__lxstat64(3, *path*, *stat_buf*) shall behave as lstat(*path*, *stat_buf*) as specified by Large File Support.

__fxstat64(3, *fildes*, *stat_buf*) shall behave as fstat(*fildes*, *stat_buf*) as specified by Large File Support.

__xstat64, __lxstat64, and __fxstat64 are not in the source standard; they are only in the binary standard.

stat64, lstat64, and fstat64 are not in the binary standard; they are only in the source standard.

_environ

Name

_environ — alias for environ - user environment

Synopsis

```
extern char **_environ;
```

Description

_environ is an alias for environ - user environment.

_nl_msg_cat_cntr

Name

_nl_msg_cat_cntr — new catalog load counter

Synopsis

```
#include <libintl.h>

extern int _nl_msg_cat_cntr;
```

Description

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

_obstack_begin

Name

_obstack_begin — initialize an obstack for use

Synopsis

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

Description

_obstack_begin initializes an obstack for use.

Future Directions

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

_obstack_newchunk

Name

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

Synopsis

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

Description

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

Future Directions

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

_sys_errlist

Name

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

Synopsis

```
#include <stdio.h>

extern const char *const _sys_errlist[];
```

Description

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

_sys_siglist

Name

_sys_siglist — array containing the names of the signal names

Synopsis

```
#include <signal.h>

```

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

Description

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

The `_sys_siglist` array is only in the binary standard; it is not in the source standard. Applications wishing to access the names of signals should use the `strsignal` function.

acct

Name

`acct` — switch process accounting on or off

Synopsis

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

Description

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

Return Value

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

Errors

ENOSYS

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

ENOMEM

: Out of memory.

EPERM

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

EACCES

: *`filename`* is not a regular file.

EIO

: Error writing to the *`filename`*.

EUSERS

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

adjtime

Name

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

Synopsis

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

Description

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

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

The adjtime is restricted to the super-user.

Return Value

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

Errors

EFAULT

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

EPERM

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

adjtimex

Name

adjtimex — tune kernel clock (DEPRECATED)

Synopsis

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

Description

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

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

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

```
  struct timex {
            int  modes;          /* mode selector */
            long offset;         /* time offset (usec) */
            long freq;           /* frequency offset (scaled ppm) */
            long maxerror;       /* maximum error (usec) */
            long esterror;       /* estimated error (usec) */
            int  status;         /* clock command/status */
            long constant;       /* pll time constant */
            long precision;      /* clock precision (usec) (read only) */
            long tolerance;      /* clock frequency tolerance (ppm)
                                           (read only) */
            struct timeval time; /* current time (read only) */
            long tick;           /* usecs between clock ticks */
  };
```

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

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

```
  #define ADJ_OFFSET_SINGLESHOT 0x8001  /* old-fashioned adjtime */
```

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

Return Value

On success, `adjtimex` returns the clock state:

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

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

Errors

EFAULT

buf does not point to writable memory.

EPERM

`buf.mode` is nonzero and the user is not super-user.

EINVAL

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

asprintf

Name

`asprintf` — write formatted output to a dynamically allocated string

Synopsis

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

Description

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

Return Value

Refer to `fprintf`.

Errors

Refer to `fprintf`.

bind_textdomain_codeset

Name

`bind_textdomain_codeset` — specify encoding for message retrieval

Synopsis

```
#include <libintl.h>
char * bind_textdomain_codeset (const char * domainname , const char * codeset );
```

Description

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

Each successive call to `bind_textdomain_codeset` function overrrides the settings made by the preceding call with the same *`domainname`*.

The `bind_textdomain_codeset` function shall return a pointer to a string containing the name of the selected codeset. The string shall be allocated internally in the function and shall not be changed or freed by the user.

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

Parameters

domainname

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

codeset

The name of the output codeset for the selected domain, or NULL to select the current codeset.

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

Return Value

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

Errors

ENOMEM

Insufficient memory available to allocate return value.

See Also

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

bindresvport

Name

`bindresvport` — bind socket to privileged IP port

Synopsis

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

Description

If the process has appropriate privilege, the `bindresvport` function shall bind a socket to a privileged IP port.

Return Value

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

Errors

`EPERM`

The process did not have appropriate privilege.

`EPFNOSUPPORT`

Address of *`sin`* did not match address family of *`sd`*.

bindtextdomain

Name

`bindtextdomain` — specify the location of a message catalog

Synopsis

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

Description

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

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

If *`dirname`* is not `NULL`, the base directory for message catalogs belonging to domain *`domainname`* shall be set to *`dirname`*. If *`dirname`* is `NULL`, the base directory for message catalogs shall not be altered.

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

`dirname` can be an absolute or relative pathname.

> Applications that wish to use `chdir` should always use absolute pathnames to avoid misadvertently selecting the wrong or non-existant directory.

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

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

Return Value

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

Errors

`ENOMEM`

> Insufficient memory was available.

See Also

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

cfmakeraw

Name

`cfmakeraw` — get and set terminal attributes

Synopsis

```
#include <termios.h>
void cfmakeraw(struct termios *termios_p);
```

Description

The `cfmakeraw` function shall set the attributes of the termios structure referenced by *`termios_p`* as follows:

```
  termios_p->c_iflag &= ~(IGNBRK|BRKINT|PARMRK|ISTRIP
                          |INLCR|IGNCR|ICRNL|IXON);

  termios_p->c_oflag &= ~OPOST;

  termios_p->c_lflag &= ~(ECHO|ECHONL|ICANON|ISIG|IEXTEN);

  termios_p->c_cflag &= ~(CSIZE|PARENB);

  termios_p->c_cflag |= CS8;
```

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

```
  tcflag_t c_iflag;      /* input modes */
  tcflag_t c_oflag;      /* output modes */
  tcflag_t c_cflag;      /* control modes */
  tcflag_t c_lflag;      /* local modes */
  cc_t c_cc[NCCS];       /* control chars */
```

cfsetspeed

Name

`cfsetspeed` — set terminal input and output data rate

Synopsis

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

Description

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

Getting and Setting the Baud Rate

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

```
  #define B0      0
  #define B50     50
  #define B75     75
  #define B110    110
  #define B134    134
  #define B150    150
  #define B200    200
  #define B300    300
  #define B600    600
  #define B1200   1200
  #define B1800   1800
  #define B2400   2400
  #define B4800   4800
  #define B9600   9600
  #define B19200  19200
  #define B38400  38400
  #ifndef _POSIX_SOURCE
  #define EXTA    19200
  #define EXTB    38400
```

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

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

Return Value

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

Errors

EINVAL

Invalid *`speed`* argument

creat

Name

`creat` — open a file

Description

`creat` is as specified in ISO POSIX (2003), but with differences as listed below.

May return ENODEV in place of ENXIO

Where the ISO POSIX (2003) specifies an ENXIO return, the implementation may return either ENXIO or ENODEV. Implementations are encouraged to return ENXIO.

> 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.

daemon

Name

daemon — run in the background

Synopsis

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

Description

The daemon function shall create a new process, detached from the controlling terminal. If successful, the calling process shall exit and the new process shall continue to execute the application in the background. If *nochdir* evaluates to true, the current directory shall not be changed. Otherwise, daemon shall change the current working directory to the root (`/'). If *noclose* evaluates to true the standard input, standard output, and standard error file descriptors shall not be altered. Otherwise, daemon shall close the standard input, standard output and standard error file descriptors and reopen them attached to /dev/null.

Return Value

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

dcgettext

Name

dcgettext — perform domain and category specific lookup in message catalog

Synopsis

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

```
#include <locale.h>
char *dcgettext(const char *domainname, const char *msgid, int category);
```

Description

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

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

Return Value

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

Errors

`dcgettext` shall not modify the `errno` global variable.

See Also

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

dcngettext

Name

`dcngettext` — perform domain and category specific lookup in message catalog with plural

Synopsis

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

```
#include <locale.h>
char *dcngettext(const char *domainname, const char *msgid1, const char *msgid2, unsigned
long int n, int category);
```

Description

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

Return Value

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

Errors

dcngettext shall not modify the errno global variable.

See Also

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

dgettext

Name

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

Synopsis

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

Description

`dgettext` is a domain specified version of `gettext`.

Parameters

domainname

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

msgid

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

Return Value

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

Errors

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

See Also

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

dngettext

Name

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

Synopsis

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

Description

`dngettext` shall be equivalent to a call to

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

See `dgettext` for more information.

See Also

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

err

Name

`err` — display formatted error messages

Synopsis

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

Description

The `err` function shall display a formatted error message on the standard error stream. First, `err` shall write the last component of the program name, a colon character, and a space character. If *`fmt`* is non-NULL, it shall be used as a format string for the `printf` family of functions, and `err` shall write the formatted message, a colon character, and a space. Finally, the error message string affiliated with the current value of the global variable `errno` shall be written, followed by a newline character.

The `err` function shall not return, the program shall terminate with the exit value of *`eval`*.

See Also

`error`, `errx`

Return Value

None.

Errors

None.

error

Name

`error` — print error message

Synopsis

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

Description

`error` shall print a message to standard error.

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

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

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

See Also

`err`, `errx`

errx

Name

errx — display formatted error message and exit

Synopsis

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

Description

The errx function shall display a formatted error message on the standard error stream. The last component of the program name, a colon character, and a space shall be output. If *fmt* is non-NULL, it shall be used as the format string for the printf family of functions, and the formatted error message, a colon character, and a space shall be output. The output shall be followed by a newline character.

errx does not return, but shall exit with the value of *eval*.

Return Value

None.

Errors

None.

See Also

error, err

fcntl

Name

fcntl — file control

Description

fcntl is as specified in ISO POSIX (2003), but with differences as listed below.

Implementation may set O_LARGEFILE

According to the *Single UNIX Specification*, only an application sets fcntl flags, for example *O_LARGEFILE*. However, this specification also allows an implementation to set *O_LARGEFILE* in the case where the system default behavior matches the *O_LARGEFILE* behavior, for example if sizeof(off_t) is 8. Thus, calling fcntl with the *F_GETFL* command may return *O_LARGEFILE* as well as flags explicitly set by the application.

fflush_unlocked

Name

`fflush_unlocked` — non thread safe fflush

Description

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

fgetwc_unlocked

Name

`fgetwc_unlocked` — non thread safe fgetwc

Description

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

flock

Name

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

Synopsis

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

Description

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

LOCK_SH

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

LOCK_EX

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

LOCK_UN

: Unlock.

LOCK_NB

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

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

Return Value

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

Errors

EWOULDBLOCK

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

fopen

Name

`fopen` — open a file

Description

`fopen` is as specified in ISO POSIX (2003), but with differences as listed below.

May return ENODEV in place of ENXIO

Where the ISO POSIX (2003) specifies an ENXIO return, the implementation may return either ENXIO or ENODEV. Implementations are encouraged to return ENXIO.

> 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.

freopen

Name

`freopen` — open a file

Description

`freopen` is as specified in ISO POSIX (2003), but with differences as listed below.

May return ENODEV in place of ENXIO

Where the ISO POSIX (2003) specifies an ENXIO return, the implementation may return either ENXIO or ENODEV. Implementations are encouraged to return ENXIO.

> 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.

getdomainname

Name

getdomainname — get NIS domain name (DEPRECATED).

Synopsis

```
#include <unistd.h>
int getdomainname (char * name , size_t namelen );
```

Description

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 *namelen*. If the NIS domain name is not currently set, getdomainname shall copy the string "(none)" to the *name*. If *namelen* is less the length of the string to be copied, getdomainname may either truncate the string to *namelen* 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 *namelen* 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

```
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

getloadavg — get system load averages

Synopsis

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

Description

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

getopt

Name

getopt — parse command line options

Synopsis

```
#include <unistd.h>
int getopt(int argc, char * const argv[], const char *optstring);

extern char *optarg;
```

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

Description

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

Argument Ordering

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

PERMUTE

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

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

REQUIRE_ORDER

The arguments in *`argv`* are processed in exactly the order given, and option processing stops when the first non-option argument is reached, or when the element of argv is "--". This ordering can be enforced either by setting the environment variable `POSIXLY_CORRECT`, or by setting the first character of *`optstring`* to '+'.

RETURN_IN_ORDER

The order of arguments is not altered, and all arguments are processed. Non-option arguments (operands) are handled as if they were the argument to an option with the value 1 ('\001'). This ordering is selected by setting the first character of *`optstring`* to '-';

Option Characteristics

LSB specifies that:

- an element of *`argv`* that starts with "-" (and is not exactly "-" or "--") is an option element.
- characters of an option element, aside from the initial "-", are option characters.

POSIX specifies that:

- applications using `getopt` shall obey the following syntax guidelines:
  - option name is a single alphanumeric character from the portable character set
  - option is preceded by the '-' delimiter character
  - options without option-arguments should be accepted when grouped behind one '-' delimiter
  - each option and option-argument is a separate argument
  - option-arguments are not optional
  - all options should precede operands on the command line
  - the argument "--" is accepted as a delimiter indicating the end of options and the consideration of subsequent 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

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`.

  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

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 ':'.
- '?' 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

LSB specifies that:

- if the variable `POSIXLY_CORRECT` is set, option processing stops as soon as a non-option argument is encountered.

- the variable _[PID]_GNU_nonoption_argv_flags_ (where [PID] is 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 be so.

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 has_arg;
            int *flag;
            int val;
};
```

The fields in this structure have the following meaning:

name

The name of the long option.

has_arg

One of:

`argument` (or 0) if the option does not take an argument,
`uired_argument` (or 1) if the option requires an argument, or
`ional_argument` (or 2) if the option takes an optional argument.

`flag`

specifies how results are returned for a long option. If flag is `NULL`, then `getopt_long` shall return *`val`*. (For example, the calling program may set val to the equivalent short option character.) Otherwise, `getopt_long` returns 0, and *`flag`* shall point to a variable which shall be set to *`val`* if the option is found, but left unchanged if the option is not found.

`val`

The value to return, or to load into the variable pointed to by flag.

Return Value

`getopt_long` returns the option character if a short option was found successfully, or ":" if there was a missing parameter for one of the options, or "?" for an unknown option character, or -1 for the end of the option list.

For a long option, `getopt_long` returns *`val`* if *`flag`* is `NULL`, and 0 otherwise. Error and -1 returns are the same as for `getopt`, plus "?" for an ambiguous match or an extraneous parameter.

getopt_long_only

Name

`getopt_long_only` — parse command line options

Synopsis

```
#define _GNU_SOURCE
#include <getopt.h>
int getopt_long_only(int argc, char * const argv[], const char *optstring, (const struct
option *longopts), int *longindex);
```

Description

`getopt_long_only` is like `getopt_long`, but "-" as well as "--" can indicate a long option. If an option that starts with "-" (not "--") doesn't match a long option, but does match a short option, it is parsed as a short option instead.

Return Value

`getopt_long_only` returns the option character if the option was found successfully, or ":" if there was a missing parameter for one of the options, or "?" for an unknown option character, or -1 for the end of the option list.

`getopt_long_only` also returns the option character when a short option is recognized. For a long option, they return val if flag is `NULL`, and 0 otherwise. Error and -1 returns are the same as for `getopt`, plus "?" for an ambiguous match or an extraneous parameter.

gettext

Name

`gettext` — Search message catalogs for a string

Synopsis

```
#include <libintl.h>
char *gettext(const char *msgid);
```

Description

The `gettext` function shall search the currently selected message catalogs for a string identified by the string *`msgid`*. If a string is located, that string shall be returned.

The `gettext` function is equivalent to `dcgettext(NULL, msgid, LC_MESSAGES)`.

Return Value

If a string is found in the currently selected message catalogs for *`msgid`*, then a pointer to that string shall be returned. Otherwise, a pointer to *`msgid`* shall be returned.

Applications shall not modify the string returned by `gettext`.

Errors

None.

The `gettext` function shall not modify `errno`.

See Also

dgettext, ngettext, dngettext, dcgettext, dcngettext, textdomain, bindtextdomain, bind_textdomain_codeset

getutent

Name

`getutent` — access user accounting database entries

Synopsis

```
#include <utmp.h>
struct utmp *getutent(void);
```

Description

The `getutent` function shall read the next entry from the user accounting database.

Return Value

Upon successful completion, `getutent` shall return a pointer to a utmp structure containing a copy of the requested entry in the user accounting database. Otherwise, a null pointer shall be returned. The return value may point to a static area which is overwritten by a subsequent call to `getutent`.

Errors

None defined.

getutent_r

Name

`getutent_r` — access user accounting database entries

Synopsis

```
int getutent_r(struct utmp * buffer, struct utmp ** result);
```

Description

The `getutent_r` function is a reentrant version of the `getutent` function. On entry, *`buffer`* should point to a user supplied buffer to which the next entry in the database will be copied, and *`result`* should point to a location where the result will be stored.

Return Value

On success, `getutent_r` shall return 0 and set the location referenced by *`result`* to a pointer to *`buffer`*. Otherwise, `getutent_r` shall return `-1` and set the location referenced by *`result`* to `NULL`.

glob64

Name

glob64 — find pathnames matching a pattern (Large File Support)

Synopsis

```
#include <glob.h>
int glob64(const char *pattern, int flags, int (*errfunc) (const char *, int), glob64_t
*pglob);
```

Description

The glob64 function is a large-file version of the glob defined in ISO POSIX (2003). It shall search for pathnames matching *pattern* according to the rules used by the shell, /bin/sh. No tilde expansion or parameter substitution is done; see wordexp.

The results of a glob64 call are stored in the structure pointed to by *pglob*, which is a glob64_t declared in glob.h with the following members:

```
typedef struct
{
  size_t gl_pathc;
  char **gl_pathv;
  size_t gl_offs;
  int gl_flags;
  void (*gl_closedir) (void *);
  struct dirent64 *(*gl_readdir64) (void *);
  void *(*gl_opendir) (const char *);
  int (*gl_lstat) (const char *, struct stat *);
  int (*gl_stat) (const char *, struct stat *);
}
```

```
glob64_t;
```

Structure members with the same name as corresponding members of a `glob_t` as defined in ISO POSIX (2003) shall have the same purpose.

Other members are defined as follows:

`gl_flags`

reserved for internal use

`gl_closedir`

pointer to a function capable of closing a directory opened by *`gl_opendir`*

`gl_readdir64`

pointer to a function capable of reading entries in a large directory

`gl_opendir`

pointer to a function capable of opening a large directory

`gl_stat`

pointer to a function capable of returning file status for a large file

`gl_lstat`

pointer to a function capable of returning file status information for a large file or symbolic link

A large file or large directory is one with a size which cannot be represented by a variable of type off_t.

Return Value

On success, 0 is returned. Other possible returns are:

GLOB_NOSPACE

out of memory

GLOB_ABORTED

read error

GLOB_NOMATCH

no match found

globfree64

Name

`globfree64` — free memory from glob64() (Large File Support)

Synopsis

```
#include <glob.h>
void globfree64(glob64_t *pglob);
```

Description

`globfree64` frees the dynamically allocated storage from an earlier call to `glob64`.

`globfree64` is a 64-bit version of `globfree`.

initgroups

Name

`initgroups` — initialize the supplementary group access list

Synopsis

```
#include <grp.h>
```

```
#include <sys/types.h>
int initgroups(const char *user, gid_t group);
```

Description

If the process has appropriate privilege, the `initgroups` function shall initialize the Supplementary Group IDs for the current process by reading the group database and using all groups of which *`user`* is a member. The additional group *`group`* is also added to the list.

Return Value

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

Errors

EPERM

: The calling process does not have sufficient privileges.

ENOMEM

: Insufficient memory to allocate group information structure.

See Also

`setgroups`

ioctl

Name

`ioctl` — control device

Synopsis

```
#include <sys/ioctl.h>
int ioctl (int d , int request , ... );
```

Description

The `ioctl` function shall manipulate the underlying device parameters of special files. *`d`* shall be an open file descriptor referring to a special file. The `ioctl` function shall take three parameters; the type and value of the third parameter is dependent on the device and *`request`*.

Conforming LSB applications shall not call `ioctl` except in situations explicitly stated in this specification.

Return Value

On success, 0 is returned. An `ioctl` may use the return value as an output parameter and return a non-negative value on success. On error, -1 is returned and the global variable `errno` is set appropriately.

Errors

EBADF

`d` is not a valid descriptor.

EFAULT

The third parameter references an inaccessible memory area.

ENOTTY

`d` is not associated with a character special device.

ENOTTY

The specified request does not apply to the kind of object that *`d`* references.

EINVAL

`request` or the third parameter is not valid.

sockio

Name

`sockio` — socket ioctl commands

Synopsis

```
#include <sys/socket.h>
#include <net/if.h>
```

```
#include <netinet/in.h>
int ioctl(int sockfd, int request, char *argp);
```

Description

Socket `ioctl` commands are a subset of the `ioctl` calls, which can perform a variety of functions on sockets. *`sockfd`* shall contain the value of a file descriptor that was created with the `socket` or `accept` calls.

Socket `ioctl` commands apply to the underlying network interfaces, and affect the entire system, not just the file descriptor used to issue the `ioctl`.

The following values for *`request`* are accepted:

SIOCGIFCONF

Gets the interface configuration list for the system.

> SIOCGIFCONF is similar to the `if_nameindex` family found in the ISO POSIX (2003) or the `getifaddrs` family found in BSD derived systems.

`argp` shall point to a ifconf structure, as described in `<net/if.h>`. Before calling, the caller shall set the *`ifc_ifcu.ifcu_req`* field to point to an array of ifreq structures, and set *`ifc_len`* to the size in bytes of this allocated array. Upon return, *`ifc_len`* will contain the size in bytes of the array which was actually used. If it is the same as the length upon calling, the caller should assume that the array was too small and try again with a larger array.

On success, SIOCGIFCONF can return any nonnegative value.

> **Rationale**
>
> Historical UNIX systems disagree on the meaning of the return value.

SIOCGIFFLAGS

Gets the interface flags for the indicated interface. *`argp`* shall point to a ifreq structure. Before calling, the caller should fill in the *`ifr_name`* field with the interface name, and upon return, the *`ifr_ifru.ifru_flags`* field is set with the interface flags.

SIOCGIFADDR

Gets the interface address for the given interface. *`argp`* shall point to a ifreq structure. Before calling, the caller should fill in the *`ifr_name`* field with the interface name, and upon return, the *`ifr_ifru.ifru_addr`* field is set with the interface address.

SIOCGIFNETMASK

Gets the network mask for the given interface. *`argp`* shall point to a ifreq structure. Before calling, the caller should fill in the *`ifr_name`* field with the interface name, and upon return, the *`ifr_ifru.ifru_netmask`* field is set with the network mask.

FIONREAD

Returns the amount of queued unread data in the receive buffer. *`argp`* shall point to an integer where the result is to be placed.

Return Value

On success, if *request* is SIOCGIFCONF, a non-negative integer shall be returned. If request is not SIOCGIFCONF, on success 0 is returned. On error, -1 is returned and the global variable errno is set appropriately.

Errors

EBADF

sockfd is not a valid descriptor.

EFAULT

argp references an inaccessible memory area.

ENOTTY

The specified *request* does not apply to the kind of object that the descriptor *sockfd* references.

EINVAL

Either *request* or *argp* is invalid.

ENOTCONN

The operation is only defined on a connected socket, but the socket wasn't connected.

kill

Name

kill — send a signal

Synopsis

```
#include <signal.h>
int kill(pid_t pid, int sig);
```

Description

kill is as specified in the *ISO POSIX (2003)*, but with differences as listed below.

Process ID -1 doesn't affect calling process

If *pid* is specified as -1, *sig* shall not be sent to the calling process. Other than this, the rules in the *ISO POSIX (2003)* apply.

Rationale

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

mbsnrtowcs

Name

`mbsnrtowcs` — convert a multibyte string to a wide character string

Synopsis

```
#include <wchar.h>
size_t mbsnrtowcs(wchar_t *dest, const char **src, size_t nms, size_t len, mbstate_t *ps);
```

Description

`mbsnrtowcs` is like `mbsrtowcs`, except that the number of bytes to be converted, starting at *`src`*, is limited to *`nms`*.

If *`dest`* is not a `NULL` pointer, `mbsnrtowcs` converts at most *`nms`* bytes from the multibyte string *`src`* to a wide-character string starting at *`dest`*. At most, *`len`* wide characters are written to *`dest`*. The state *`ps`* is updated.

The conversion is effectively performed by repeatedly calling:

```
mbrtowc(dest, *src, n, ps)
```

where *n* is some positive number, as long as this call succeeds, and then incrementing *dest* by one and *src* by the number of bytes consumed.

The conversion can stop for three reasons:

- An invalid multibyte sequence has been encountered. In this case *src* is left pointing to the invalid multibyte sequence, (size_t)(-1) is returned, and `errno` is set to EILSEQ.
- The *nms* limit forces a stop, or *len* non-L'\0' wide characters have been stored at *dest*. In this case, *src* is left pointing to the next multibyte sequence to be converted, and the number of wide characters written to *dest* is returned.
- The multibyte string has been completely converted, including the terminating '\0' (which has the side effect of bringing back *ps* to the initial state). In this case, *src* is set to `NULL`, and the number of wide characters written to *dest*, excluding the terminating L'\0' character, is returned.

If *dest* is `NULL`, *len* is ignored, and the conversion proceeds as above, except that the converted wide characters are not written out to memory, and that no destination length limit exists.

In both of the above cases, if *ps* is a `NULL` pointer, a static anonymous state only known to `mbsnrtowcs` is used instead.

The programmer shall ensure that there is room for at least *len* wide characters at *dest*.

Return Value

`mbsnrtowcs` returns the number of wide characters that make up the converted part of the wide character string, not including the terminating null wide character. If an invalid multibyte sequence was encountered, (size_t)(-1) is returned, and the global variable `errno` is set to EILSEQ.

Notes

The behavior of `mbsnrtowcs` depends on the `LC_CTYPE` category of the current locale.

Passing NULL as *ps* is not multi-thread safe.

memmem

Name

`memmem` — locate bytes

Synopsis

```
#define _GNU_SOURCE
```

```
#include <string.h>
void *memmem(const void *haystack, size_t haystacklen, const void *needle, size_t
needlelen);
```

Description

memmem finds the start of the first occurrence of the byte array referenced by *needle* of length *needlelen* in the memory area *haystack* of length *haystacklen*.

Return Value

memmem returns a pointer to the beginning of the byte array, or NULL if the byte array is not found.

Notes

Earlier versions of the C library (prior to glibc 2.1) contained a memmem with various problems, and application developers should treat this function with care.

memrchr

Name

memrchr — scan memory for a character

Synopsis

```
#include <string.h>
void *memrchr(const void *s, int c, size_t n);
```

Description

The memrchr function shall locate the last occurence of *c* (converted to an unsigned char) in the initial *n* bytes (each interpreted as an unsigned char) of the object pointed to by *s*.

Return Value

The memrchr shall return a pointer to the located byte, or a null pointer if the byte does not occur in the object.

Errors

No errors are defined.

See Also

memchr

ngettext

Name

ngettext — Search message catalogs for plural string

Synopsis

```
#include <libintl.h>
char *ngettext(const char *msgid1, const char *msgid2, unsigned long int n);
```

Description

The ngettext function shall search the currently selected message catalogs for a string matching the singular string *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 on *n*) of the string shall be returned.

The ngettext function is equivalent to dcngettext(NULL, msgid1, msgid2, n, LC_MESSAGES).

Return Value

If a string is found in the currently selected message catalogs for *msgid1*, then if *n* is 1 a pointer to the located string shall be returned. If *n* is not 1, a pointer to an appropriately pluralized version of the string shall be returned. If no message could be found in the currently selected mesage catalogs, then if *n* is 1, a pointer to *msgid1* shall be returned, otherwise a pointer to *msgid2* shall be returned.

Applications shall not modify the string returned by ngettext.

Errors

None.

The ngettext function shall not modify errno.

See Also

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

obstack_free

Name

obstack_free — free an object in the obstack

Synopsis

```
#include <obstack.h>
void obstack_free((struct obstack *obstack), void *block);
```

Description

obstack_free frees an object in the obstack.

Future Directions

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

open

Name

open — open a file

Synopsis

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

Description

The open function shall behave as specified in *ISO POSIX (2003)*, except with differences as listed below.

May return ENODEV in place of ENXIO

Where ISO POSIX (2003) specifies an ENXIO return, a conforming implementation may return either ENXIO or ENODEV. Implementations are encouraged to return ENXIO.

Rationale

As of spring 2004, no Linux kernel patches to switch to ENXIO are known, but it is believed that such a kernel patch would be accepted if submitted.

opterr

Name

`opterr` — external variable used in getopt()

Synopsis

```
extern int opterr;
```

Description

`opterr` is used as a flag to suppress an error message generated by `getopt`. When `opterr` is set to `0`, it suppresses the error message generated by `getopt` when that function does not recognize an option character.

optind

Name

`optind` — external variable used in getopt()

Synopsis

```
extern int optind;
```

Description

`optind` holds the current index of the array `argv`[], which contains the command line options being parsed by `getopt`.

optopt

Name

`optopt` — external variable used in getopt()

Synopsis

```
extern int optopt;
```

Description

`optopt` holds the unknown option character when that option character is not recognized by `getopt`.

pmap_getport

Name

`pmap_getport` — Find the port number assigned to a service registered with a portmapper.

Synopsis

```
#include <pmap_clnt.h>
u_short *pmap_getport(struct sockaddr_in *address, __const u_long program, __const u_long
*version, u_int protocol);
```

Description

The `pmap_getport` function shall return the port number assigned to a service registered with a RPC Binding service running on a given target system, using the protocol described in RFC 1833: Binding Protocols for ONC RPC Version 2. The `pmap_getport` function shall be called given the RPC program number *`program`*, the program version *`version`*, and transport protocol *`protocol`*. Conforming implementations shall support both `IPPROTO_UDP` and `IPPROTO_TCP` protocols. On entry, *`address`* shall specify the address of the system on which the portmapper to be contacted resides. The value of `address->sin_port` shall be ignored, and the standard value for the portmapper port shall always be used.

> Security and network restrictions may prevent a conforming application from contacting a remote RPC Binding Service.

Return Value

On success, the `pmap_getport` function shall return the port number in host byte order of the RPC application registered with the remote portmapper. On failure, if either the program was not registered or the remote portmapper service could not be reached, the `pmap_getport` function shall return 0. If the remote portmap service could not be reached, the status is left in the global variable `rpc_createerr`.

pmap_set

Name

pmap_set — Establishes mapping to machine's RPC Bind service.

Synopsis

```
#include <rpc/pmap_clnt.h>
 *pmap_set(__const u_long program, __const u_long version, int protocol, u_short port);
```

Description

pmap_set establishes a mapping between the triple *[program,version,protocol]* and *port* on the machine's RPC Bind service. The value of *protocol* is most likely IPPROTO_UDP or IPPROTO_TCP. Automatically done by svc_register.

Return Value

pmap_set returns 1 if it suceeds, 0 otherwise.

pmap_unset

Name

pmap_unset — Destroys RPC Binding

Synopsis

```

#include <rpc/rpc.h>

void pmap_unset(u_long progNum, u_long versNum);
```

Description

As a user interface to the RPC Bind service, pmap_unset destroys all mapping between the triple [*progNum*,*versNum*, *] and ports on the machine's RPC Bind service.

Return Value

pmap_unset returns 1 if it succeeds, zero otherwise.

psignal

Name

`psignal` — print signal message

Synopsis

```
#include <signal.h>
void psignal(int sig, const char *s);

extern const char *const sys_siglist[]
```

Description

The `psignal` function shall display a message on the `stderr` stream. If *`s`* is not the null pointer, 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 string describing the signal number *`sig`*; otherwise `psignal` shall display only a message describing the signal number *`sig`*. If *`sig`* is invalid, the message displayed shall indicate an unknown signal.

The array `sys_siglist` holds the signal description strings indexed by signal number.

Return Value

`psignal` returns no value.

random_r

Name

`random_r` — generate random number

Synopsis

```
int random_r((struct random_data *__restrict __buf), int32_t *__restrict __result);
```

Description

`random_r` is a reentrant version of `random`, which generates a pseudorandom number.

Future Directions

Since this function requires support from other functions not specified in this specification (most notably `initstate_r`), a future version of this specification may deprecate this interface.

setbuffer

Name

`setbuffer` — stream buffering operation

Synopsis

```
#include <stdio.h>
void setbuffer(FILE *stream, char *buf, size_t size);
```

Description

`setbuffer` is an alias for the call to `setvbuf`. It works the same, except that the size of the buffer in `setbuffer` is up to the caller, rather than being determined by the default *`BUFSIZ`*.

setdomainname

Name

`setdomainname` — set NIS domain name (DEPRECATED).

Synopsis

```
#include <unistd.h>
int setdomainname (char * name , size_t namelen );
```

Description

If NIS is in use, set the NIS domain name. Note that this is not the same as the domain name which provides the domain portion of a fully qualified domain name (for example, in DNS). If NIS is not in use, this function may set the domain name anyway, or it may fail.

This call shall fail unless the caller has appropriate privileges.

`namelen` shall be the length of the string pointed to by *`name`*.

Return Value

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

Errors

`EPERM`

: The process did not have sufficient privilege to set the domain name.

`EINVAL`

: *`name`* is a null pointer.

setgroups

Name

`setgroups` — set list of supplementary group IDs

Synopsis

```
#include <grp.h>
int setgroups(size_t size, const gid_t *list);
```

Description

If the process has appropriate privilege, the `setgroups` function shall set the supplementary group IDs for the current process. *`list`* shall reference an array of *`size`* group IDs. A process may have at most `NGROUPS_MAX` supplementary group IDs.

Return Value

On successful completion, 0 is returned. On error, -1 is returned and the `errno` is set to indicate the error.

Errors

EFAULT

: *`list`* has an invalid address.

EPERM

: The process does not have appropriate privileges.

EINVAL

: *`size`* is greater than `NGROUPS_MAX`.

sethostid

Name

`sethostid` — set the unique identifier of the current host

Synopsis

```
#include <unistd.h>
int sethostid(long int hostid);
```

Description

`sethostid` sets a unique 32-bit identifier for the current machine. The 32-bit identifier is intended to be unique among all UNIX systems in existence. This normally resembles the Internet address for the local machine as returned by `gethostbyname`(3), and thus usually never needs to be set.

The `sethostid` call is restricted to the superuser.

`hostid` is stored in the file `/etc/hostid`.

Return Value

`gethostid` returns the 32-bit identifier for the current host as set by `sethostid`(2).

Files

`/etc/hostid`

sethostname

Name

`sethostname` — set host name

Synopsis

```
#include <unistd.h>
#include <sys/param.h.h>
```

```
#include <sys/utsname.h>
int sethostname(const char *name, size_t len);
```

Description

If the process has appropriate privileges, the sethostname function shall change the host name for the current macine. The *name* shall point to a null-terminated string of at most *len* bytes that holds the new hostname.

If the symbol HOST_NAME_MAX is defined, or if sysconf(_SC_HOST_NAME_MAX) returns a value greater than 0, this value shall represent the maximum length of the new hostname. Otherwise, if the symbol MAXHOSTLEN is defined, this value shall represent the maximum length for the new hostname. If none of these values are defined, the maximum length shall be the size of the *nodename* field of the utsname structure.

Return Value

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

Errors

EINVAL

len is negative or larger than the maximum allowed size.

EPERM

the process did not have appropriate privilege.

EFAULT

name is an invalid address.

Rationale

ISO POSIX (2003) guarantees that:

> Maximum length of a host name (not including the terminating null) as returned from the gethostname function shall be at least 255 bytes.

The glibc C library does not currently define HOST_NAME_MAX, and although it provides the name _SC_HOST_NAME_MAX a call to sysconf returns -1 and does not alter errno in this case (indicating that there is no restriction on the hostname length). However, the glibc manual idicates that some implementations may have MAXHOSTNAMELEN as a means of detecting the maximum length, while the Linux kernel at release 2.4 and 2.6 stores this hostname in the utsname structure. While the glibc manual suggests simply shortening the name until sethostname succeeds, the LSB requires that one of the first four mechanisms works. Future versions of glibc may provide a more reasonable result from sysconf(_SC_HOST_NAME_MAX).

setsockopt

Name

`setsockopt` — set options on sockets

Synopsis

```
#include <sys/socket.h>
#include <netinet/in.h>
int setsockopt(int sockfd, int level, int optname, void *optval, socklen_t optlen);
```

Description

In addition to the `setsockopt` options specified in SUSv3, `setsockopt` also supports the options specified here.

The following `setsockopt` operations are provided for `level` IPPROTO_IP:

IP_MULTICAST_TTL

Set or reads the time-to-live value of outgoing multicast packets for this socket. *`optval`* is a pointer to an integer which contains the new TTL value.

IP_MULTICAST_LOOP

Sets a boolean flag indicating whether multicast packets originating locally should be looped back to the local sockets. *`optval`* is a pointer to an integer which contains the new flag value.

IP_ADD_MEMBERSHIP

Join a multicast group. *`optval`* is a pointer to a ip_mreq structure. Before calling, the caller should fill in the *`imr_multiaddr`* field with the multicast group address and the *`imr_address`* field with the address of the local interface. If *`imr_address`* is set to INADDR_ANY, then an appropriate interface is chosen by the system.

IP_DROP_MEMBERSHIP

Leave a multicast group. *`optval`* is a pointer to a ip_mreq structure containing the same values as were used with IP_ADD_MEMBERSHIP.

IP_MULTICAST_IF

Set the local device for a multicast socket. *`optval`* is a pointer to a ip_mreq structure initialized in the same manner as with IP_ADD_MEMBERSHIP.

The ip_mreq structure contains two struct in_addr fields: *`imr_multiaddr`* and *`imr_address`*.

Return Value

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

setutent

Name

`setutent` — access user accounting database entries

Synopsis

```
#include <utmp.h>
void setutent(void);
```

Description

The `setutent` function shall reset the user accounting database such that the next call to `getutent` shall be return the first record in the database. It is recommended to call it before any of the other functions that operate on the user accounting databases (e.g. `getutent`)

Return Value

None.

sigandset

Name

`sigandset` — build a new signal set by combining the two input sets using logical AND

Synopsis

```
#include <signal.h>
int sigandset(sigset_t *set, const sigset_t *left, const sigset_t *right);
```

Description

The `sigandset` shall combine the two signal sets referenced by *`left`* and *`right`*, using a logical AND operation, and shall place the result in the location referenced by *`set`*, The resulting signal set shall contain only signals that are in both the set referenced by *`left`* and the set referenced by *`right`*.

Return Value

On success, `sigandset` shall return 0. Otherise, `sigandset` shall return -1 and set `errno` to indicate the error.

Errors

`EINVAL`

One or more of *`set`*, *`left`*, or *`right`* was a null pointer.

See Also

`sigorset`

sigblock

Name

`sigblock` — manipulate the signal mask

Synopsis

```
#include _BSD_SOURCE
```

```
#include <signal.h>
int sigblock(int mask);
```

Description

The `sigblock` function shall add the signals corresponding to the bits set in *`mask`* to the set of signals currently being blocked from delivery.

Return Value

The `sigblock` function shall return the previous signal mask.

Errors

None.

Notes

`sigblock` is made obsolete by `sigprocmask`(2). A future version of this specification may deprecate this function.

siggetmask

Name

`siggetmask` — manipulate the signal mask

Synopsis

```
#define _BSD_SOURCE
#include <signal.h>
int siggetmask(void);
```

Description

The `siggetmask` function shall return the current set of masked signals.

Notes

`siggetmask` is made obsolete by `sigprocmask`(2).

sigisemptyset

Name

`sigisemptyset` — check for empty signal set

Synopsis

```
#include <signal.h>
int sigisemptyset(const sigset_t *set);
```

Description

The `sigisemptyset` function shall check for empty signal set referenced by *`set`*.

Return Value

The `sigisemptyset` function shall return a positive non-zero value if the signal set referenced by *`set`* is empty, or zero if this set is empty. On error, `sigisemptyset` shall return -1 and set `errno` to indicate the error.

Errors

`EINVAL`

`set` is a null pointer.

sigorset

Name

`sigorset` — build a new signal set by combining the two input sets using logical OR

Synopsis

```
#include <signal.h>
int sigorset(sigset_t *set, const sigset_t *left, const sigset_t *right);
```

Description

The `sigorset` shall combine the two signal sets referenced by *`left`* and *`right`*, using a logical OR operation, and shall place the result in the location referenced by *`set`*, The resulting signal set shall contain only signals that are in either the set referenced by *`left`* or the set referenced by *`right`*.

Return Value

On success, `sigorset` shall return 0. Otherise, `sigorset` shall return -1 and set `errno` to indicate the error.

Errors

`EINVAL`

One or more of *`set`*, *`left`*, or *`right`* was a null pointer.

See Also

`sigorset`

sigreturn

Name

`sigreturn` — return from signal handler and cleanup stack frame

Synopsis

```
int sigreturn(unsigned long __unused);
```

Description

The `sigreturn` function is used by the system to cleanup after a signal handler has returned. This function is not in the source standard; it is only in the binary standard.

Return Value

`sigreturn` never returns.

stime

Name

`stime` — set time

Synopsis

```
#define _SVID_SOURCE
```

```
#include <time.h>
int stime(time_t *t);
```

Description

If the process has appropriate privilege, the `stime` function shall set the system's idea of the time and date. Time, referenced by *t*, is measured in seconds from the epoch (defined in ISO POSIX (2003) as 00:00:00 UTC January 1, 1970).

Return Value

On success, `stime` shall return 0. Otherwise, `stime` shall return -1 and `errno` shall be set to indicate the error.

Errors

`EPERM`

The process does not have appropriate privilege.

`EINVAL`

t is a null pointer.

stpcpy

Name

`stpcpy` — copy a string returning a pointer to its end

Synopsis

```
#include <string.h>
char *stpcpy(char * restrict dest, const char * restrict src);
```

Description

The `stpcpy` function shall copy the string pointed to by *src* (including the terminating '\0' character) to the array pointed to by *dest*. The strings may not overlap, and the destination string *dest* shall be large enough to receive the copy.

Return Value

`stpcpy` returns a pointer to the end of the string *dest* (that is, the address of the terminating '\0' character) rather than the beginning.

Example

This program uses `stpcpy` to concatenate `foo` and `bar` to produce `foobar`, which it then prints.

```
  #include <string.h>
```

```
  int
  main (void)
  {
    char buffer[256];
    char *to = buffer;
    to = stpcpy (to, "foo");
    to = stpcpy (to, "bar");
    printf ("%s\n", buffer);
  }
```

stpncpy

Name

`stpncpy` — copy a fixed-size string, returning a pointer to its end

Synopsis

```
#include <string.h>
char *stpncpy(char * restrict dest, const char * restrict src, size_t n);
```

Description

The `stpncpy` function shall copy at most *`n`* characters from the string pointed to by *`src`*, including the terminating \0 character, to the array pointed to by *`dest`*. Exactly *`n`* characters are written at *`dest`*. If the length `strlen`*`(src)`* is smaller than *`n`*, the remaining characters in *`dest`* are filled with \0 characters. If the length `strlen`*`(src)`* is greater than or equal to *`n`*, *`dest`* will not be \0 terminated.

The strings may not overlap.

The programmer shall ensure that there is room for at least *`n`* characters at *`dest`*.

Return Value

The `stpncpy` function shall return a pointer to the terminating NULL in *`dest`*, or, if *`dest`* is not NULL-terminated, *`dest`* + *`n`*.

strcasestr

Name

`strcasestr` — locate a substring ignoring case

Synopsis

```
#include <string.h>
char *strcasestr(const char *s1, const char *s2);
```

Description

The `strcasestr` shall behave as `strstr`, except that it shall ignore the case of both strings. The `strcasestr` function shall be locale aware; that is `strcasestr` shall behave as if both strings had been converted to lower case in the current locale before the comparison is performed.

Return Value

Upon successful completion, `strcasestr` shall return a pointer to the located string or a null pointer if the string is not found. If *`s2`* points to a string with zero length, the function shall return *`s1`*.

strerror_r

Name

`strerror_r` — reentrant version of strerror

Synopsis

```
#include <string.h>
char *strerror_r(int errnum, char *buf, size_t buflen);
```

Description

`strerror_r` is a reentrant version of `strerror`. `strerror_r` returns a pointer to an error message corresponding to error number *`errnum`*. The returned pointer may point within the buffer *`buf`* (at most *`buflen`* bytes).

> Note the optional use of the buffer, unlike the `strerror_r` found in ISO POSIX (2003), in which the message is always copied into the supplied buffer. The return types also differ.

strfry

Name

`strfry` — randomize a string

Synopsis

```
#include <string.h>
char *strfry(char *string);
```

Description

`strfry` randomizes the contents of *`string`* by using `rand`(3) to randomly swap characters in the string. The result is an anagram of *`string`*.

Return Value

`strfry` returns a pointer to the randomized string.

strndup

Name

`strndup` — return a malloc'd copy of at most the specified number of bytes of a string

Synopsis

```
#include <string.h>
char *strndup(const char *string, size_t n);
```

Description

The `strndup` function shall return a `malloc`'d copy of at most *`n`* bytes of *`string`*. The resultant string shall be terminated even if no NULL terminator appears before *`string`*+*`n`*.

Return Value

On success, `strndup` shall return a pointer to a newly allocated block of memory containing a copy of at most *`n`* bytes of *`string`*. Otherwise, `strndup` shall return NULL and set `errno` to indicate the error.

Errors

`ENOMEM`

Insufficient memory available.

strnlen

Name

`strnlen` — determine the length of a fixed-size string

Synopsis

```
#include <string.h>
size_t strnlen(const char *s, size_t maxlen);
```

Description

`strnlen` returns the number of characters in the string *`s`*, not including the terminating \0 character, but at most *`maxlen`*. In doing this, `strnlen` looks only at the first *`maxlen`* characters at *`s`* and never beyond *`s`* + *`maxlen`*.

Return Value

`strnlen` returns `strlen`*`(s)`*, if that is less than *`maxlen`*, or *`maxlen`* if there is no \0 character among the first *`maxlen`* characters pointed to by *`s`*.

strptime

Name

`strptime` — parse a time string

Description

The `strptime` shall behave as specified in the *ISO POSIX (2003)* with differences as listed below.

Number of leading zeroes may be limited

The *ISO POSIX (2003)* specifies fields for which "leading zeros are permitted but not required"; however, applications shall not expect to be able to supply more leading zeroes for these fields than would be implied by the range of the field. Implementations may choose to either match an input with excess leading 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` are acceptable inputs, but inputs such as `0000`, `0366` and the like are not.

Rationale

glibc developers consider it appropriate behavior to forbid excess leading zeroes. When trying to parse a given input against several format strings, forbidding excess leading zeroes could be helpful. For example, if one matches `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 perverse to parse that date as November 12, year 26. The second pattern parses it as December 26, year 11.

The *ISO POSIX (2003)* is not explicit that an unlimited number of leading zeroes are required, although it may imply this. The LSB explicitly allows implementations to have either behavior. Future versions of this standard may require implementations to forbid excess leading zeroes.

An Interpretation Request is currently pending against ISO POSIX (2003) for this matter.

strsep

Name

`strsep` — extract token from string

Synopsis

```
#include <string.h>
char *strsep(char **stringp, const char *delim);
```

Description

The `strsep` function shall find the first token in the string referenced by the pointer *`stringp`*, using the characters in *`delim`* as delimiters.

If *`stringp`* is NULL, `strsep` shall return NULL and do nothing else.

If *`stringp`* is non-NULL, `strsep` shall find the first token in the string referenced by *`stringp`*, where tokens are delimited by characters in the string *`delim`*. This token shall be terminated with a \0 character by overwriting the delimiter, and *`stringp`* shall be updated to point past the token. In case no delimiter was found, the token is taken to be the entire string referenced by *`stringp`*, and the location referenced by *`stringp`* is made NULL.

Return Value

`strsep` shall return a pointer to the beginning of the token.

Notes

The `strsep` function was introduced as a replacement for `strtok`, since the latter cannot handle empty fields. However, `strtok` conforms to ISO C (1999) and to ISO POSIX (2003) and hence is more portable.

See Also

`strtok`, `strtok_r`.

strsignal

Name

`strsignal` — return string describing signal

Synopsis

```
#define _GNU_SOURCE
```

```
#include <string.h>
char *strsignal(int sig);

extern const char * const sys_siglist[];
```

Description

The `strsignal` function shall return a pointer to a string describing the signal number *`sig`*. The string can only be used until the next call to `strsignal`.

The array `sys_siglist` holds the signal description strings indexed by signal number. This array should not be accessed directly by applications.

Return Value

If *`sig`* is a valid signal number, `strsignal` shall return a pointer to the appropriate description string. Otherwise, `strsignal` shall return either a pointer to the string `"unknown signal"`, or a null pointer.

Although the function is not declared as returning a pointer to a constant character string, applications shall not modify the returned string.

strtoq

Name

`strtoq` — convert string value to a long or quad_t integer

Synopsis

```
#include <sys/types.h>
#include <stdlib.h>
```

```
#include <limits.h>
quadt strtoq(const char *nptr, char **endptr, int base);
```

Description

`strtoq` converts the string *`nptr`* to a quadt value. The conversion is done according to the given base, which shall be between `2` and `36` inclusive, or be the special value `0`.

`nptr` may begin with an arbitrary amount of white space (as determined by `isspace`(3)), followed by a single optional + or - sign character. If *`base`* is `0` or `16`, the string may then include a 0x prefix, and the number will be read 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 (octal).

The remainder of the string is converted to a long value in the obvious manner, stopping at the first character which is not a valid digit in the given base. (In bases above 10, the letter `A` in either upper or lower case represents 10, `B` represents 11, and so forth, with `Z` representing 35.)

Return Value

`strtoq` returns the result of the conversion, unless the value would underflow or overflow. If an underflow occurs, `strtoq` returns `QUAD_MIN`. If an overflow occurs, `strtoq` returns `QUAD_MAX`. In both cases, the global variable `errno` is set to ERANGE.

Errors

ERANGE

The given string was out of range; the value converted has been clamped.

strtouq

Name

`strtouq` — convert a string to an uquad_t

Synopsis

```
#include <sys/types.h>
#include <stdlib.h>
```

```
#include <limits.h>
uquadt strtouq(const char *nptr, char **endptr, int base);
```

Description

`strtouq` converts the string *`nptr`* to a uquadt value. The conversion is done according to the given base, which shall be between `2` and `36` inclusive, or be the special value `0`.

`nptr` may begin with an arbitrary amount of white space (as determined by `isspace`(3)), followed by a single optional + or - sign character. If *`base`* is `0` or `16`, the string may then include a 0x prefix, and the number will be read 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 (octal).

The remainder of the string is converted to an unsigned long value in the obvious manner, stopping at the end of the 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 either upper or lower case represents 10, `B` represents 11, and so forth, with `Z` representing 35.)

Return Value

On success, `strtouq` returns either the result of the conversion or, if there was a leading minus sign, the negation of the result of the conversion, unless the original (non-negated) value would overflow. In the case of an overflow the function returns `UQUAD_MAX` and the global variable `errno` is set to ERANGE.

Errors

ERANGE

: The given string was out of range; the value converted has been clamped.

strverscmp

Name

`strverscmp` — compare strings holding name and indices/version numbers

Synopsis

```
#include <string.h>
int strverscmp(const char *s1, const char *s2);
```

Description

The `strversmp` function shall compare two strings in a similar manner to `strcmp`. If *`s1`* and *`s2`* contain no digits, `strversmp` shall behave as `strcmp`.

The strings are compared by scanning from left to right. If a digit or sequence of digits is encountered in both strings at the same position, the digit sequence is specially compared, as described below. If the digit sequences compared equal, the string comparison resumes in both *`s1`* and *`s2`* after the digit sequence.

Digit sequences are classified as either "integral" or "fractional". A fractional digit sequence begins with a `'0'`; otherwise the digit sequence shall be treated as an integral digit sequence.

If two integral digit sequences are encountered, they shall be compared as integers for equality. A fractional digit sequence shall always compare less than an integral digit sequence. If two fractional digit sequences are being compared, then if the common prefix contains only leading zeroes, the longer part shall compare less than the shorter; otherwise the comparison shall be strictly numeric.

Examples

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

svc_register — Register Remote Procedure Call Interface

Synopsis

```
#include <rpc/rpc.h>
void svc_register(SVCXPRT *xprt, u_long prognum, u_long versnum, void (*dispatch)(), u_long
protocol);
```

Description

The svc_register function shall associate the program identified by *prognum* at version *versnum* with the service dispatch procedure, *dispatch*. If *protocol* is zero, the service is not registered with the portmap service. If *protocol* is non-zero, then a mapping of the triple [*prognum*, *versnum*, *protocol*] to xprt->xp_port is established with the local portmap service. The procedure *dispatch* has the following form:

```
int dispatch(struct svc_req * request, SVCXPRT * xprt);
```

Return Value

svc_register returns 1 if it succeeds, and zero otherwise.

svc_run

Name

svc_run — Waits for RPC requests to arrive and calls service procedure.

Synopsis

```
#include <rpc/svc.h>
void svc_run(void);
```

Description

The svc_run function shall wait for RPC requests to arrive, read and unpack each request, and dispatch it to the appropriate registered handler. Under normal conditions, svc_run shall not return; it shall only return if serious errors occur that prevent further processing.

svc_sendreply

Name

`svc_sendreply` — called by RPC service's dispatch routine

Synopsis

```
svc_sendreply(SVCXPRT *xprt, xdrproc_t outproc, char out);
```

Description

Called by an RPC service's dispatch routine to send the results of a remote procedure call. The parameter *`xprt`* is the request's associated transport handle; *`outproc`* is the XDR routine which is used to encode the results; and *`out`* is the address of the results. This routine returns one if it succeeds, zero other-wise.

svctcp_create

Name

`svctcp_create` — Creates a TCP/IP-based RPC service transport.

Synopsis

```
#include <rpc/rpc.h>
SVCXPRT *svctcp_create(int sock, u_int send_buf_size, u_int recv_buf_size);
```

Description

`svctcp_create` cretes a TCP/IP-based RPC service transport, to which it returns a pointer. The transport is associated with the socket *`sock`*, which may be `RPC_ANYSOCK`, in which case a new socket is created. If the socket is not bound to a local TCP port, ten this routine binds it to an arbitrary port. Upon completion, `xprt->xp_sock` is the transport's socket descriptor, and `xprt->xp_port` is the transport's port number. Since TCP-based RPC uses buffered I/O, users may specify the size of buffers; values of zero choose suitable defaults.

Return Value

`svctcp_create` returns NULL if it fails, or a pointer to the RPC service transport otherwise.

svcudp_create

Name

`svcudp_create` — Creates a UDP-based RPC service transport.

Synopsis

```
SVCXPRT *
svcudp_create(int sock);
```

Description

This call is equivalent to `svcudp_bufcreate` *`(sock, SZ, SZ)`* for some default size SZ.

system

Name

`system` — execute a shell command

Synopsis

```
#include <stdlib.h>
int system(const char *string);
```

Description

The `system` function shall behave as described in ISO POSIX (2003).

Notes

The fact that `system` ignores interrupts is often not what a program wants. ISO POSIX (2003) describes some of the consequences; an additional consequence is that a program calling `system` from a loop cannot be reliably interrupted. Many programs will want to use the `exec` family of functions instead.

Do not use `system` from a program with `suid` or `sgid` privileges, because unexpected values for some environment variables might be used to subvert system integrity. Use the `exec` family of functions instead, but not `execlp` or `execvp`. `system` will not, in fact, work properly from programs with `suid` or `sgid` privileges on systems on which `/bin/sh` is **bash** version 2, since **bash** 2 drops privileges on startup. (Debian uses a modified **bash** which does not do this when invoked as **sh**.)

The check for the availability of `/bin/sh` is not actually performed; it is always assumed to be available. ISO C (1999) specifies the check, but ISO POSIX (2003) specifies that the return shall always be nonzero, since a system without the shell is not conforming, and it is this that is implemented.

It is possible for the shell command to return 127, so that code is not a sure indication that the `execve` call failed; check the global variable `errno` to make sure.

textdomain

Name

`textdomain` — set the current default message domain

Synopsis

```
#include <libintl.h>
char *textdomain(const char *domainname);
```

Description

The `textdomain` function shall set the current default message domain to *`domainname`*. Subsequent calls to `gettext` and `ngettext` use the default message domain.

If *`domainname`* is NULL, the default message domain shall not be altered.

If *`domainname`* is "", `textdomain` shall reset the default domain to the system default of "messages".

Return

On success, `textdomain` shall return the currently selected domain. Otherwise, a null pointer shall be returned, and `errno` set to indicate the error.

Errors

`ENOMEM`

: Insufficent memory available.

unlink

Name

`unlink` — remove a directory entry

Synopsis

```
int unlink(const char *path);
```

Description

`unlink` is as specified in ISO POSIX (2003), but with differences as listed below.

See also Additional behaviors: unlink/link on directory.

May return EISDIR on directories

If *`path`* specifies a directory, the implementation may return EISDIR instead of EPERM as specified by ISO POSIX (2003).

Rationale

The Linux kernel has deliberately chosen EISDIR for this case and does not expect to change (Al Viro, personal communication).

vasprintf

Name

`vasprintf` — write formatted output to a dynamically allocated string

Synopsis

```
#include <stdarg.h>
```

```
#include <stdio.h>
int vasprintf(char ** restrict ptr, const char * restrict format, va_list arg);
```

Description

The `vasprintf` function shall write formatted output to a dynamically allocated string, and store the address of that string in the location referenced by *`ptr`*. It shall behave as `asprintf`, except that instead of being called with a variable number of arguments, it is called with an argument list as defined by `<stdarg.h>`.

Return Value

Refer to `fprintf`.

Errors

Refer to `fprintf`.

vdprintf

Name

`vdprintf` — write formatted output to a file descriptor

Synopsis

```
#include <stdio.h>
int vdprintf(int fd, const char * restrict format, va_list arg);
```

Description

The `vdprintf` shall behave as `vfprintf`, except that the first argument is a file descriptor rather than a STDIO stream.

Return Value

Refer to `fprintf`.

Errors

Refer to `fprintf`.

verrx

Name

verrx — display formatted error message and exit

Synopsis

```
#include <stdarg.h>
#include <err.h>
void verrx(int eval, const char *fmt, va_list args);
```

Description

The verrx shall behave as errx except that instead of being called with a variable number of arguments, it is called with an argument list as defined by <stdarg.h>.

verrx does not return, but exits with the value of *eval*.

Return Value

None.

Errors

None.

vsyslog

Name

vsyslog — log to system log

Synopsis

```
#include <stdarg.h>
#include <syslog.h>
void vsyslog(int priority, char *message, va_list arglist);
```

Description

The vsyslog function is identical to syslog as specified in ISO POSIX (2003), except that *arglist* (as defined by stdarg.h) replaces the variable number of arguments.

wait3

Name

`wait3` — wait for child process

Description

`wait3` is as specified in the SUSv2 but with differences as listed below.

WCONTINUED and WIFCONTINUED optional

Implementations need not support the functionality of WCONTINUED or WIFCONTINUED.

wait4

Name

`wait4` — wait for process termination, BSD style

Synopsis

```
#include <sys/types.h>
#include <sys/resource.h>
```

```
#include <sys/wait.h>
pid_t wait4(pid_t pid, int *status, int options, (struct rusage *rusage));
```

Description

`wait4` suspends execution of the current process until a child (as specified by *`pid`*) has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child (as requested by *`pid`*) has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed.

The value of *`pid`* can be one of:

< -1

: wait for any child process whose process group ID is equal to the absolute value of *`pid`*.

-1

: wait for any child process; this is equivalent to calling `wait3`.

0

: wait for any child process whose process group ID is equal to that of the calling process.

> 0

: wait for the child whose process ID is equal to the value of *`pid`*.

The value of options is a bitwise or of zero or more of the following constants:

WNOHANG

: return immediately if no child is there to be waited for.

WUNTRACED

: return for children that are stopped, and whose status has not been reported.

If status is not NULL, `wait4` stores status information in the location *`status`*. This status can be evaluated with the following macros:

> These macros take the `status` value (an `int`) as an argument -- not a pointer to the value!

WIFEXITED(status)

: is nonzero if the child exited normally.

WEXITSTATUS(status)

: evaluates to the least significant eight bits of the return code of the child that terminated, which may have been set as the argument to a call to `exit` or as the argument for a return statement in the main program. This macro can only be evaluated if `WIFEXITED` returned nonzero.

WIFSIGNALED(status)

: returns true if the child process exited because of a signal that was not caught.

WTERMSIG(status)

returns the number of the signal that caused the child process to terminate. This macro can only be evaluated if `WIFSIGNALED` returned nonzero.

WIFSTOPPED(status)

returns true if the child process that caused the return is currently stopped; this is only possible if the call was done using `WUNTRACED`.

WSTOPSIG(status)

returns the number of the signal that caused the child to stop. This macro can only be evaluated if `WIFSTOPPED` returned nonzero.

If *`rusage`* is not NULL, the struct rusage (as defined in `sys/resource.h`) that it points to will be filled with accounting information. (See `getrusage`(2) for details.

Return Value

On success, the process ID of the child that exited is returned. On error, -1 is returned (in particular, when no unwaited-for child processes of the specified kind exist), or 0 if `WNOHANG` was used and no child was available yet. In the latter two cases, the global variable `errno` is set appropriately.

Errors

ECHILD

No unwaited-for child process as specified does exist.

ERESTARTSYS

A `WNOHANG` was not set and an unblocked signal or a `SIGCHILD` was caught. This error is returned by the system call. The library interface is not allowed to return ERESTARTSYS, but will return EINTR.

waitpid

Name

`waitpid` — wait for child process

Description

`waitpid` is as specified in ISO POSIX (2003), but with differences as listed below.

Need not support WCONTINUED or WIFCONTINUED

Implementations need not support the functionality of `WCONTINUED` or `WIFCONTINUED`.

warn

Name

warn — formatted error messages

Synopsis

```
#include <err.h>
void warn(const char *fmt ...);
```

Description

The warn function shall display a formatted error message on the standard error stream. The output shall consist of the last component of the program name, a colon character, and a space character. If *fmt* is non-NULL, it shall be used as a format string for the printf family of functions, and the formatted message, a colon character, and a space are written to stderr. Finally, the error message string affiliated with the current value of the global variable errno shall be written to stderr, followed by a newline character.

Return Value

None.

Errors

None.

warnx

Name

warnx — formatted error messages

Synopsis

```
#include <err.h>
void warnx(const char *fmt ...);
```

Description

The warnx function shall display a formatted error message on the standard error stream. The last component of the program name, a colon character, and a space shall be output. If *fmt* is non-NULL, it shall be used as the format string for the printf family of functions, and the formatted error message, a colon character, and a space shall be output. The output shall be followed by a newline character.

Return Value

None.

Errors

None.

wcpcpy

Name

wcpcpy — copy a wide character string, returning a pointer to its end

Synopsis

```
#include <wchar.h>
wchar_t *wcpcpy(wchar_t *dest, const wchar_t *src);
```

Description

wcpcpy is the wide-character equivalent of stpcpy. It copies the wide character string *src*, including the terminating L'\0' character, to the array *dest*.

The strings may not overlap.

The programmer shall ensure that there is room for at least wcslen*(src)*+1 wide characters at *dest*.

Return Value

wcpcpy returns a pointer to the end of the wide-character string *dest*, that is, a pointer to the terminating L'\0' character.

wcpncpy

Name

wcpncpy — copy a fixed-size string of wide characters, returning a pointer to its end

Synopsis

```
#include <wchar.h>
wchar_t *wcpncpy(wchar_t *dest, const wchar_t *src, size_t n);
```

Description

wcpncpy is the wide-character equivalent of stpncpy. It copies at most *n* wide characters from the wide-character string *src*, including the terminating L'\0' character, to the array *dest*. Exactly *n* wide characters are written at *dest*. If the length wcslen*(src)* is smaller than *n*, the remaining wide characters in the array *dest* are filled with L'\0' characters. If the length wcslen*(src)* is greater than or equal to *n*, the string *dest* will not be L'\0' terminated.

The strings may not overlap.

The programmer shall ensure that there is room for at least *n* wide characters at *dest*.

Return Value

wcpncpy returns a pointer to the wide character one past the last non-null wide character written.

wcscasecmp

Name

wcscasecmp — compare two wide-character strings, ignoring case

Synopsis

```
#include <wchar.h>
int wcscasecmp(const wchar_t *s1, const wchar_t *s2);
```

Description

wcscasecmp is the wide-character equivalent of strcasecmp. It compares the wide-character string *s1* and the wide-character string *s2*, ignoring case differences (towupper, towlower).

Return Value

wcscasecmp returns 0 if the wide-character strings *s1* and *s2* are equal except for case distinctions. It returns a positive integer if *s1* is greater than *s2*, ignoring case. It returns a negative integer if *s1* is smaller than *s2*, ignoring case.

Notes

The behavior of wcscasecmp depends upon the LC_CTYPE category of the current locale.

wcsdup

Name

wcsdup — duplicate a wide-character string

Synopsis

```
#include <wchar.h>
wchar_t *wcsdup(const wchar_t *s);
```

Description

wcsdup is the wide-character equivalent of strdup. It allocates and returns a new wide-character string whose initial contents is a duplicate of the wide-character string *s*.

Memory for the new wide-character string is obtained with malloc(3), and can be freed with free(3).

Return Value

wcsdup returns a pointer to the new wide-character string, or NULL if sufficient memory was not available.

wcsncasecmp

Name

`wcsncasecmp` — compare two fixed-size wide-character strings, ignoring case

Synopsis

```
#include <wchar.h>

int wcsncasecmp(const wchar_t *s1, const wchar_t *s2, size_t n);
```

Description

`wcsncasecmp` is the wide-character equivalent of `strncasecmp`. It compares the wide-character string *`s1`* and the wide-character string *`s2`*, but at most *`n`* wide characters from each string, ignoring case differences (towupper, towlower).

Return Value

`wcscasecmp` returns 0 if the wide-character strings *`s1`* and *`s2`*, truncated to at most length *`n`*, are equal except for case distinctions. It returns a positive integer if truncated *`s1`* is greater than truncated *`s2`*, ignoring case. It returns a negative integer if truncated *`s1`* is smaller than truncated *`s2`*, ignoring case.

Notes

The behavior of `wcsncasecmp` depends upon the `LC_CTYPE` category of the current locale.

wcsnlen

Name

wcsnlen — determine the length of a fixed-size wide-character string

Synopsis

```
#include <wchar.h>
size_t wcsnlen(const wchar_t *s, size_t maxlen);
```

Description

wcsnlen is the wide-character equivalent of strnlen. It returns the number of wide-characters in the string *s*, not including the terminating L'\0' character, but at most *maxlen*. In doing this, wcsnlen looks only at the first *maxlen* wide-characters at *s* and never beyond *s* + *maxlen*.

Return Value

wcsnlen returns wcslen*(s)* if that is less than *maxlen*, or *maxlen* if there is no L'\0' character among the first *maxlen* wide characters pointed to by *s*.

Notes

The behavior of wcsncasecmp depends on the LC_CTYPE category of the current locale.

wcsnrtombs

Name

wcsnrtombs — convert a wide character string to a multi-byte string

Synopsis

```
#include <wchar.h>
size_t wcsnrtombs(char *dest, const wchar_t **src, size_t nwc, size_t len, mbstate_t *ps);
```

Description

wcsnrtombs is like wcsrtombs, except that the number of wide characters to be converted, starting at *src*, is limited to *nwc*.

If *dest* is not a NULL pointer, wcsnrtombs converts at most *nwc* wide characters from the wide-character string *src* to a multibyte string starting at *dest*. At most *len* bytes are written to *dest*. The state *ps* is updated.

The conversion is effectively performed by repeatedly calling:

```
wcrtomb(dest, *src, ps)
```

as long as this call succeeds, and then incrementing *dest* by the number of bytes written and *src* by 1.

The conversion can stop for three reasons:

- A wide character has been encountered that cannot be represented as a multibyte sequence (according to the current locale). In this case *src* is left pointing to the invalid wide character, (size_t)(-1) is returned, and errno is set to EILSEQ.
- *nws* wide characters have been converted without encountering a L'\0', or the length limit forces a stop. In this case, *src* is left pointing to the next wide character to be converted, and the number bytes written to *dest* is returned.
- The wide-character string has been completely converted, including the terminating L'\0' (which has the side effect of bringing back *ps* to the initial state). In this case, *src* is set to NULL, and the number of bytes written to *dest*, excluding the terminating L'\0' byte, is returned.

If *dest* is NULL, *len* is ignored, and the conversion proceeds as above, except that the converted bytes are not written out to memory, and that no destination length limit exists.

In both of the above cases, if *ps* is a NULL pointer, a static anonymous state only known to wcsnrtombs is used instead.

The programmer shall ensure that there is room for at least *len* bytes at *dest*.

Return Value

wcsnrtombs returns the number of bytes that make up the converted part of multibyte sequence, not including the terminating L'\0' byte. If a wide character was encountered which could not be converted, (size_t)(-1) is returned, and the global variable errno set to EILSEQ.

Notes

The behavior of `wcsnrtombs` depends on the `LC_CTYPE` category of the current locale.

Passing NULL as *`ps`* is not multi-thread safe.

wcstoq

Name

`wcstoq` — convert wide string to long long int representation

Synopsis

```
#include <wchar.h>
long long int wcstoq(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);
```

Description

The `wcstoq` function shall convert the initial portion of the wide string *`nptr`* to `long long int` representation. It is identical to `wcstoll`.

Return Value

Refer to `wcstoll`.

Errors

Refer to `wcstoll`.

wcstouq

Name

wcstouq — convert wide string to unsigned long long int representation

Synopsis

```
#include <wchar.h>
unsigned long long int wcstouq(const wchar_t * restrict nptr, wchar_t ** restrict endptr,
int base);
```

Description

The wcstouq function shall convert the initial portion of the wide string *nptr* to unsigned long long int representation. It is identical to wcstoull.

Return Value

Refer to wcstoull.

Errors

Refer to wcstoull.

xdr_u_int

Name

xdr_u_int — library routines for external data representation

Synopsis

```
int xdr_u_int(XDR * xdrs, unsigned int * up);
```

Description

xdr_u_int is a filter primitive that translates between C unsigned integers and their external representations.

Return Value

On success, 1 is returned. On error, 0 is returned.

1.5. Interfaces for libm

Table 1-29 defines the library name and shared object name for the libm library

Table 1-29. libm Definition

Library:	libm
SONAME:	See archLSB.

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

ISO C (1999)
SUSv2
ISO POSIX (2003)

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 1-30, with the full functionality as described in the referenced underlying specification.

Table 1-30. libm - Math Function Interfaces

acos [1]	cexp [1]	expf [1]	jnf [2]	remquof [1]
acosf [1]	cexpf [1]	expl [1]	jnl [2]	remquol [1]
acosh [1]	cexpl [1]	expm1 [1]	ldexp [1]	rint [1]
acoshf [1]	cimag [1]	fabs [1]	ldexpf [1]	rintf [1]
acoshl [1]	cimagf [1]	fabsf [1]	ldexpl [1]	rintl [1]
acosl [1]	cimagl [1]	fabsl [1]	lgamma [1]	round [1]
asin [1]	clog [1]	fdim [1]	lgamma_r [2]	roundf [1]
asinf [1]	clog10 [2]	fdimf [1]	lgammaf [1]	roundl [1]
asinh [1]	clog10f [2]	fdiml [1]	lgammaf_r [2]	scalb [1]
asinhf [1]	clog10l [2]	feclearexcept [1]	lgammal [1]	scalbf [2]
asinhl [1]	clogf [1]	fegetenv [1]	lgammal_r [2]	scalbl [2]
asinl [1]	clogl [1]	fegetexceptflag [1]	llrint [1]	scalbln [1]
atan [1]	conj [1]	fegetround [1]	llrintf [1]	scalblnf [1]
atan2 [1]	conjf [1]	feholdexcept [1]	llrintl [1]	scalblnl [1]
atan2f [1]	conjl [1]	feraiseexcept [1]	llround [1]	scalbn [1]
atan2l [1]	copysign [1]	fesetenv [1]	llroundf [1]	scalbnf [1]
atanf [1]	copysignf [1]	fesetexceptflag [1]	llroundl [1]	scalbnl [1]
atanh [1]	copysignl [1]	fesetround [1]	log [1]	significand [2]
atanhf [1]	cos [1]	fetestexcept [1]	log10 [1]	significandf [2]

atanhl [1]	cosf [1]	feupdateenv [1]	log10f [1]	significandl [2]
atanl [1]	cosh [1]	finite [3]	log10l [1]	sin [1]
cabs [1]	coshf [1]	finitef [2]	log1p [1]	sincos [2]
cabsf [1]	coshl [1]	finitel [2]	logb [1]	sincosf [2]
cabsl [1]	cosl [1]	floor [1]	logf [1]	sincosl [2]
cacos [1]	cpow [1]	floorf [1]	logl [1]	sinf [1]
cacosf [1]	cpowf [1]	floorl [1]	lrint [1]	sinh [1]
cacosh [1]	cpowl [1]	fma [1]	lrintf [1]	sinhf [1]
cacoshf [1]	cproj [1]	fmaf [1]	lrintl [1]	sinhl [1]
cacoshl [1]	cprojf [1]	fmal [1]	lround [1]	sinl [1]
cacosl [1]	cprojl [1]	fmax [1]	lroundf [1]	sqrt [1]
carg [1]	creal [1]	fmaxf [1]	lroundl [1]	sqrtf [1]
cargf [1]	crealf [1]	fmaxl [1]	matherr [2]	sqrtl [1]
cargl [1]	creall [1]	fmin [1]	modf [1]	tan [1]
casin [1]	csin [1]	fminf [1]	modff [1]	tanf [1]
casinf [1]	csinf [1]	fminl [1]	modfl [1]	tanh [1]
casinh [1]	csinh [1]	fmod [1]	nan [1]	tanhf [1]
casinhf [1]	csinhf [1]	fmodf [1]	nanf [1]	tanhl [1]
casinhl [1]	csinhl [1]	fmodl [1]	nanl [1]	tanl [1]
casinl [1]	csinl [1]	frexp [1]	nearbyint [1]	tgamma [1]
catan [1]	csqrt [1]	frexpf [1]	nearbyintf [1]	tgammaf [1]
catanf [1]	csqrtf [1]	frexpl [1]	nearbyintl [1]	tgammal [1]
catanh [1]	csqrtl [1]	gamma [3]	nextafter [1]	trunc [1]
catanhf [1]	ctan [1]	gammaf [2]	nextafterf [1]	truncf [1]
catanhl [1]	ctanf [1]	gammal [2]	nextafterl [1]	truncl [1]
catanl [1]	ctanh [1]	hypot [1]	nexttoward [1]	y0 [1]
cbrt [1]	ctanhf [1]	hypotf [1]	nexttowardf [1]	y0f [2]
cbrtf [1]	ctanhl [1]	hypotl [1]	nexttowardl [1]	y0l [2]
cbrtl [1]	ctanl [1]	ilogb [1]	pow [1]	y1 [1]
ccos [1]	dremf [2]	ilogbf [1]	pow10 [2]	y1f [2]
ccosf [1]	dreml [2]	ilogbl [1]	pow10f [2]	y1l [2]

ccosh [1]	erf [1]	j0 [1]	pow10l [2]	yn [1]
ccoshf [1]	erfc [1]	j0f [2]	powf [1]	ynf [2]
ccoshl [1]	erfcf [1]	j0l [2]	powl [1]	ynl [2]
ccosl [1]	erfcl [1]	j1 [1]	remainder [1]	
ceil [1]	erff [1]	j1f [2]	remainderf [1]	
ceilf [1]	erfl [1]	j1l [2]	remainderl [1]	
ceill [1]	exp [1]	jn [1]	remquo [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. ISO C (1999)

[3]. SUSv2

An LSB conforming implementation shall provide the generic data interfaces for Math specified in Table 1-31, with the full functionality as described in the referenced underlying specification.

Table 1-31. libm - Math Data Interfaces

signgam [1]				

Referenced Specification(s)

[1]. ISO POSIX (2003)

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) ? __isinff (x): sizeof (x) == sizeof
(double) ? __isinf (x) : __isinfl (x))
#define isnan(x)      (sizeof (x) == sizeof (float) ? __isnanf (x)  : sizeof (x) == sizeof
(double) ? __isnan (x) : __isnanl (x))

#define HUGE_VAL        0x1.0p2047
#define HUGE_VALF       0x1.0p255f
#define HUGE_VALL       0x1.0p32767L

#define NAN     ((float)0x7fc00000UL)
#define M_1_PI  0.31830988618379067154
#define M_LOG10E        0.43429448190325182765
#define M_2_PI  0.63661977236758134308
#define M_LN2   0.69314718055994530942
#define M_SQRT1_2       0.70710678118654752440
#define M_PI_4  0.78539816339744830962
#define M_2_SQRTPI      1.12837916709551257390
#define M_SQRT2 1.41421356237309504880
#define M_LOG2E 1.44269504088896340740
#define M_PI_2  1.57079632679489661923
#define M_LN10  2.30258509299404568402
#define M_E     2.71828182845904523536
#define M_PI    3.14159265358979323846
#define INFINITY        HUGE_VALF

#define MATH_ERRNO      1
#define MATH_ERREXCEPT  2
```

1.7. Interfaces for libpthread

Table 1-32 defines the library name and shared object name for the libpthread library

Table 1-32. libpthread Definition

Library:	libpthread
SONAME:	libpthread.so.0

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

Large File Support
this specification

4417 ISO POSIX (2003)

1.7.1. Realtime Threads

4418 1.7.1.1. Interfaces for Realtime Threads

4419 No external functions are defined for libpthread - Realtime Threads

1.7.2. Advanced Realtime Threads

4420 1.7.2.1. Interfaces for Advanced Realtime Threads

4421 No external functions are defined for libpthread - Advanced Realtime Threads

1.7.3. Posix Threads

4422 1.7.3.1. Interfaces for Posix Threads

4423 An LSB conforming implementation shall provide the generic functions for Posix Threads specified in Table 1-33,
4424 with the full functionality as described in the referenced underlying specification.

4425 **Table 1-33. libpthread - Posix Threads Function Interfaces**

_pthread_cleanup_pop [1]	pthread_cancel [2]	pthread_join [2]	pthread_rwlock_destroy [2]	pthread_setconcurrency [2]
_pthread_cleanup_push [1]	pthread_cond_broadcast [2]	pthread_key_create [2]	pthread_rwlock_init [2]	pthread_setspecific [2]
pread [2]	pthread_cond_destroy [2]	pthread_key_delete [2]	pthread_rwlock_rdlock [2]	pthread_sigmask [2]
pread64 [3]	pthread_cond_init [2]	pthread_kill [2]	pthread_rwlock_timedrdlock [2]	pthread_testcancel [2]
pthread_attr_destroy [2]	pthread_cond_signal [2]	pthread_mutex_destroy [2]	pthread_rwlock_timedwrlock [2]	pwrite [2]
pthread_attr_getdetachstate [2]	pthread_cond_timedwait [2]	pthread_mutex_init [2]	pthread_rwlock_tryrdlock [2]	pwrite64 [3]
pthread_attr_getguardsize [2]	pthread_cond_wait [2]	pthread_mutex_lock [2]	pthread_rwlock_trywrlock [2]	sem_close [2]
pthread_attr_getschedparam [2]	pthread_condattr_destroy [2]	pthread_mutex_trylock [2]	pthread_rwlock_unlock [2]	sem_destroy [2]
pthread_attr_getstackaddr [2]	pthread_condattr_getpshared [2]	pthread_mutex_unlock [2]	pthread_rwlock_wrlock [2]	sem_getvalue [2]
pthread_attr_getstacksize [2]	pthread_condattr_init [2]	pthread_mutexattr_destroy [2]	pthread_rwlockattr_destroy [2]	sem_init [2]

pthread_attr_init [2]	pthread_condattr_setpshared [2]	pthread_mutexattr_getpshared [2]	pthread_rwlockattr_getpshared [2]	sem_open [2]
pthread_attr_setdetachstate [2]	pthread_create [2]	pthread_mutexattr_gettype [2]	pthread_rwlockattr_init [2]	sem_post [2]
pthread_attr_setguardsize [2]	pthread_detach [2]	pthread_mutexattr_init [2]	pthread_rwlockattr_setpshared [2]	sem_timedwait [2]
pthread_attr_setschedparam [2]	pthread_equal [2]	pthread_mutexattr_setpshared [2]	pthread_self [2]	sem_trywait [2]
pthread_attr_setstackaddr [2]	pthread_exit [2]	pthread_mutexattr_settype [2]	pthread_setcancelstate [2]	sem_unlink [2]
pthread_attr_setstacksize [2]	pthread_getspecific [2]	pthread_once [2]	pthread_setcanceltype [2]	sem_wait [2]

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. Large File Support

1.8. Data Definitions for libpthread

This section defines global identifiers and their values that are associated with interfaces contained in libpthread. 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.

1.8.1. pthread.h

```
#define PTHREAD_MUTEX_DEFAULT   1
#define PTHREAD_MUTEX_NORMAL    1
#define PTHREAD_MUTEX_TIMED_NP  1
#define PTHREAD_MUTEX_RECURSIVE 2
#define PTHREAD_RWLOCK_DEFAULT_NP       2
#define PTHREAD_MUTEX_ERRORCHECK        3
#define pthread_cleanup_pop(execute)    _pthread_cleanup_pop(& _buffer,(execute));}
#define __LOCK_INITIALIZER      { 0, 0 }
#define PTHREAD_RWLOCK_INITIALIZER      { __LOCK_INITIALIZER, 0, NULL, NULL,
NULL,PTHREAD_RWLOCK_DEFAULT_NP, PTHREAD_PROCESS_PRIVATE }
#define PTHREAD_MUTEX_INITIALIZER     {0,0,0,PTHREAD_MUTEX_TIMED_NP,__LOCK_INITIALIZER}
```

```
4450  #define pthread_cleanup_push(routine,arg)       {struct _pthread_cleanup_buffer
4451  _buffer;_pthread_cleanup_push(& _buffer,(routine),(arg));
4452  #define PTHREAD_COND_INITIALIZER        {__LOCK_INITIALIZER,0}
4453
4454  struct _pthread_cleanup_buffer
4455  {
4456    void (*__routine) (void *);
4457    void *__arg;
4458    int __canceltype;
4459    struct _pthread_cleanup_buffer *__prev;
4460  }
4461   ;
4462  typedef unsigned int pthread_key_t;
4463  typedef int pthread_once_t;
4464  typedef long long __pthread_cond_align_t;
4465
4466  typedef unsigned long pthread_t;
4467  struct _pthread_fastlock
4468  {
4469    long __status;
4470    int __spinlock;
4471  }
4472   ;
4473
4474  typedef struct _pthread_descr_struct *_pthread_descr;
4475
4476  typedef struct
4477  {
4478    int __m_reserved;
4479    int __m_count;
4480    _pthread_descr __m_owner;
4481    int __m_kind;
4482    struct _pthread_fastlock __m_lock;
4483  }
4484  pthread_mutex_t;
4485  typedef struct
4486  {
4487    int __mutexkind;
4488  }
4489  pthread_mutexattr_t;
4490
4491  typedef struct
4492  {
4493    int __detachstate;
4494    int __schedpolicy;
4495    struct sched_param __schedparam;
4496    int __inheritsched;
4497    int __scope;
4498    size_t __guardsize;
4499    int __stackaddr_set;
4500    void *__stackaddr;
4501    unsigned long __stacksize;
4502  }
```

```
pthread_attr_t;

typedef struct
{
  struct _pthread_fastlock __c_lock;
  _pthread_descr __c_waiting;
  char __padding[48 - sizeof (struct _pthread_fastlock) -
                 sizeof (_pthread_descr) - sizeof (__pthread_cond_align_t)];
  __pthread_cond_align_t __align;
}
pthread_cond_t;
typedef struct
{
  int __dummy;
}
pthread_condattr_t;

typedef struct _pthread_rwlock_t
{
  struct _pthread_fastlock __rw_lock;
  int __rw_readers;
  _pthread_descr __rw_writer;
  _pthread_descr __rw_read_waiting;
  _pthread_descr __rw_write_waiting;
  int __rw_kind;
  int __rw_pshared;
}
pthread_rwlock_t;
typedef struct
{
  int __lockkind;
  int __pshared;
}
pthread_rwlockattr_t;

#define PTHREAD_CREATE_JOINABLE 0
#define PTHREAD_INHERIT_SCHED   0
#define PTHREAD_ONCE_INIT       0
#define PTHREAD_PROCESS_PRIVATE 0
#define PTHREAD_CREATE_DETACHED 1
#define PTHREAD_EXPLICIT_SCHED  1
#define PTHREAD_PROCESS_SHARED  1

#define PTHREAD_CANCELED        ((void*)-1)
#define PTHREAD_CANCEL_DEFERRED 0
#define PTHREAD_CANCEL_ENABLE   0
#define PTHREAD_CANCEL_ASYNCHRONOUS     1
#define PTHREAD_CANCEL_DISABLE  1
```

1.8.2. semaphore.h

```
typedef struct
{
  struct _pthread_fastlock __sem_lock;
  int __sem_value;
  _pthread_descr __sem_waiting;
}
sem_t;
#define SEM_FAILED      ((sem_t*)0)

#define SEM_VALUE_MAX   ((int)((~0u)>>1))
```

1.9. Interface Definitions for libpthread

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

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

_pthread_cleanup_pop

Name

`_pthread_cleanup_pop` — establish cancellation handlers

Synopsis

```
#include <pthread.h>
void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *, int);
```

Description

The `_pthread_cleanup_pop` function provides an implementation of the `pthread_cleanup_pop` macro described in *ISO POSIX (2003)*.

The `_pthread_cleanup_pop` function is not in the source standard; it is only in the binary standard.

_pthread_cleanup_push

Name

_pthread_cleanup_push — establish cancellation handlers

Synopsis

```
#include <pthread.h>
void _pthread_cleanup_push(struct _pthread_cleanup_buffer *, void (*) (void *), void *);
```

Description

The _pthread_cleanup_push function provides an implementation of the pthread_cleanup_push macro described in *ISO POSIX (2003)*.

The _pthread_cleanup_push function is not in the source standard; it is only in the binary standard.

1.10. Interfaces for libgcc_s

Table 1-34 defines the library name and shared object name for the libgcc_s library

Table 1-34. libgcc_s Definition

Library:	libgcc_s
SONAME:	libgcc_s.so.1

1.10.1. Unwind Library

1.10.1.1. Interfaces for Unwind Library

No external functions are defined for libgcc_s - Unwind Library

1.11. Data Definitions for libgcc_s

This section defines global identifiers and their values that are associated with interfaces contained in libgcc_s. 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.

1.11.1. unwind.h

```
struct dwarf_eh_base
```

```
{
  void *tbase;
  void *dbase;
  void *func;
}
 ;
struct _Unwind_Context;

typedef unsigned int _Unwind_Ptr;
typedef unsigned int _Unwind_Word;

typedef enum
{
  _URC_NO_REASON, _URC_FOREIGN_EXCEPTION_CAUGHT = 1, _URC_FATAL_PHASE2_ERROR =
    2, _URC_FATAL_PHASE1_ERROR = 3, _URC_NORMAL_STOP = 4, _URC_END_OF_STACK =
    5, _URC_HANDLER_FOUND = 6, _URC_INSTALL_CONTEXT =
    7, _URC_CONTINUE_UNWIND = 8
}
_Unwind_Reason_Code;

struct _Unwind_Exception
{
  _Unwind_Exception_Class;
  _Unwind_Exception_Cleanup_Fn;
  _Unwind_Word;
  _Unwind_Word;
}
 ;
#define _UA_SEARCH_PHASE        1
#define _UA_END_OF_STACK        16
#define _UA_CLEANUP_PHASE       2
#define _UA_HANDLER_FRAME       4
#define _UA_FORCE_UNWIND        8
```

1.12. Interfaces for libdl

Table 1-35 defines the library name and shared object name for the libdl library

Table 1-35. libdl Definition

Library:	libdl
SONAME:	libdl.so.2

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

this specification
ISO POSIX (2003)

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 1-36, with the full functionality as described in the referenced underlying specification.

Table 1-36. libdl - Dynamic Loader Function Interfaces

dladdr [1]	dlclose [2]	dlerror [2]	dlopen [1]	dlsym [1]

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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 *) -1l)
#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

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

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

dladdr

Name

`dladdr` — find the shared object containing a given address

Synopsis

```
#include <dlfcn.h>

typedef struct {
             const char  *dli_fname;
             void        *dli_fbase;
             const char  *dli_sname;
             void        *dli_saddr;
```

```
} Dl_info;

int dladdr(void *addr, Dl_info *dlip);
```

Description

The `dladdr` function shall query the dynamic linker for information about the shared object containing the address *`addr`*. The information shall be returned in the user supplied data structure referenced by *`dlip`*.

The structure shall contain at least the following members:

`dli_fname`

The pathname of the shared object containing the address

`dli_fbase`

The base address at which the shared object is mapped into the address space of the calling process.

`dli_sname`

The name of the nearest runtime symbol with value less than or equal to *`addr`*. Where possible, the symbol name shall be returned as it would appear in C source code.

If no symbol with a suitable value is found, both this field and *`dli_saddr`* shall be set to `NULL`.

`dli_saddr`

The address of the symbol returned in *`dli_sname`*.

The behavior of `dladdr` is only specified in dynamically linked programs.

Return Value

On success, `dladdr` shall return non-zero, and the structure referenced by *`dlip`* shall be filled in as described. Otherwise, `dladdr` shall return zero, and the cause of the error can be fetched with `dlerr`.

Errors

See `dlerr`.

Environment

`LD_LIBRARY_PATH`

directory search-path for object files

dlopen

Name

`dlopen` — open dynamic object

Synopsis

```
#include <dlfcn.h>

void * dlopen(const char *filename, int flag);
```

Description

dlopen shall behave as specified in ISO POSIX (2003), but with additional behaviors listed below.

If the file argument does not contain a slash character, then the system shall look for a library of that name in at least the following directories, and use the first one which is found:

- The directories specified by the `DT_RPATH` dynamic entry.
- The directories specified in the `LD_LIBRARY_PATH` environment variable (which is a colon separated list of pathnames). This step shall be skipped for setuid and setgid executables.
- A set of directories sufficient to contain the libraries specified in this standard.

  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.

dlsym

Name

`dlsym` — obtain the address of a symbol from a dlopen object

Description

`dlsym` is as specified in the ISO POSIX (2003), 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 POSIX (2003).

1.15. Interfaces for libcrypt

Table 1-37 defines the library name and shared object name for the libcrypt library

Table 1-37. libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

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

ISO POSIX (2003)

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 1-38, with the full functionality as described in the referenced underlying specification.

Table 1-38. libcrypt - Encryption Function Interfaces

crypt [1]	encrypt [1]	setkey [1]		

Referenced Specification(s)

[1]. ISO POSIX (2003)

1.16. Interfaces for libpam

Table 1-39 defines the library name and shared object name for the libpam library

Table 1-39. 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. [1]

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

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 1-40, with the full functionality as described in the referenced underlying specification.

Table 1-40. libpam - Pluggable Authentication API Function Interfaces

pam_acct_mgmt [1]	pam_close_session	pam_get_item [1]	pam_set_item [1]	pam_strerror [1]

	[1]			
pam_authenticate [1]	pam_end [1]	pam_getenvlist [1]	pam_setcred [1]	
pam_chauthtok [1]	pam_fail_delay [1]	pam_open_session [1]	pam_start [1]	

Referenced Specification(s)

[1]. 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;
struct pam_message
{
  int msg_style;
  const char *msg;
}
 ;
struct pam_response
{
  char *resp;
  int resp_retcode;
}
 ;

struct pam_conv
{
  int (*conv) (int num_msg, const struct pam_message * *msg,
               struct pam_response * *resp, void *appdata_ptr);
  void *appdata_ptr;
}
 ;
#define PAM_PROMPT_ECHO_OFF     1
#define PAM_PROMPT_ECHO_ON      2
#define PAM_ERROR_MSG   3
#define PAM_TEXT_INFO   4
```

```
4768
4769    #define PAM_SERVICE     1
4770    #define PAM_USER        2
4771    #define PAM_TTY 3
4772    #define PAM_RHOST       4
4773    #define PAM_CONV        5
4774    #define PAM_RUSER       8
4775    #define PAM_USER_PROMPT 9
4776
4777    #define PAM_SUCCESS     0
4778    #define PAM_OPEN_ERR    1
4779    #define PAM_USER_UNKNOWN        10
4780    #define PAM_MAXTRIES    11
4781    #define PAM_NEW_AUTHTOK_REQD    12
4782    #define PAM_ACCT_EXPIRED        13
4783    #define PAM_SESSION_ERR 14
4784    #define PAM_CRED_UNAVAIL        15
4785    #define PAM_CRED_EXPIRED        16
4786    #define PAM_CRED_ERR    17
4787    #define PAM_CONV_ERR    19
4788    #define PAM_SYMBOL_ERR  2
4789    #define PAM_AUTHTOK_ERR 20
4790    #define PAM_AUTHTOK_RECOVER_ERR 21
4791    #define PAM_AUTHTOK_LOCK_BUSY   22
4792    #define PAM_AUTHTOK_DISABLE_AGING       23
4793    #define PAM_TRY_AGAIN   24
4794    #define PAM_ABORT       26
4795    #define PAM_AUTHTOK_EXPIRED     27
4796    #define PAM_BAD_ITEM    29
4797    #define PAM_SERVICE_ERR 3
4798    #define PAM_SYSTEM_ERR  4
4799    #define PAM_BUF_ERR     5
4800    #define PAM_PERM_DENIED 6
4801    #define PAM_AUTH_ERR    7
4802    #define PAM_CRED_INSUFFICIENT   8
4803    #define PAM_AUTHINFO_UNAVAIL    9
4804
4805    #define PAM_DISALLOW_NULL_AUTHTOK       0x0001U
4806    #define PAM_ESTABLISH_CRED      0x0002U
4807    #define PAM_DELETE_CRED 0x0004U
4808    #define PAM_REINITIALIZE_CRED   0x0008U
4809    #define PAM_REFRESH_CRED        0x0010U
4810    #define PAM_CHANGE_EXPIRED_AUTHTOK      0x0020U
4811    #define PAM_SILENT      0x8000U
```

1.18. Interface Definitions for libpam

4812 The following interfaces are included in libpam and are defined by this specification. Unless otherwise noted, these
4813 interfaces shall be included in the source standard.

4814 Other interfaces listed above for libpam shall behave as described in the referenced base document.

pam_acct_mgmt

Name

`pam_acct_mgmt` — establish the status of a user's account

Synopsis

```
#include <security/pam_appl.h>
int pam_acct_mgmt(pam_handle_t *pamh, int flags);
```

Description

`pam_acct_mgmt` establishes the account's usability and the user's accessibility to the system. It is typically called after the user has been authenticated.

`flags` may be specified as any valid flag (namely, one of those applicable to the *`flags`* argument of `pam_authenticate`). Additionally, the value of *`flags`* may be logically or'd with `PAM_SILENT`.

Return Value

PAM_SUCCESS

: Success.

PAM_NEW_AUTHTOK_REQD

: User is valid, but user's authentication token has expired. The correct response to this return-value is to require that the user satisfy the `pam_chauthtok` function before obtaining service. It may not be possible for an application to do this. In such a case, the user should be denied access until the account password is updated.

PAM_ACCT_EXPIRED

: User is no longer permitted access to the system.

PAM_AUTH_ERR

: Authentication error.

PAM_PERM_DENIED

: User is not permitted to gain access at this time.

PAM_USER_UNKNOWN

: User is not known to a module's account management component.

Errors

May be translated to text with `pam_strerror`.

pam_authenticate

Name

`pam_authenticate` — authenticate the user

Synopsis

```
#include <security/pam_appl.h>
int pam_authenticate(pam_handle_t *pamh, int flags);
```

Description

`pam_authenticate` serves as an interface to the authentication mechanisms of the loaded modules.

`flags` is an optional parameter that may be specified by the following value:

PAM_DISALLOW_NULL_AUTHTOK

: Instruct the authentication modules to return `PAM_AUTH_ERR` if the user does not have a registered authorization token.

Additionally, the value of *`flags`* may be logically or'd with `PAM_SILENT`.

The process may need to be privileged in order to successfully call this function.

Return Value

PAM_SUCCESS

: Success.

PAM_AUTH_ERR

: User was not authenticated or process did not have sufficient privileges to perform authentication.

PAM_CRED_INSUFFICIENT

: Application does not have sufficient credentials to authenticate the user.

PAM_AUTHINFO_UNAVAIL

: Modules were not able to access the authentication information. This might be due to a network or hardware failure, etc.

PAM_USER_UNKNOWN

: Supplied username is not known to the authentication service.

PAM_MAXTRIES

: One or more authentication modules has reached its limit of tries authenticating the user. Do not try again.

PAM_ABORT

: One or more authentication modules failed to load.

Errors

4862 May be translated to text with `pam_strerror`.

pam_chauthtok

Name

`pam_chauthtok` — change the authentication token for a given user

Synopsis

```
#include <security/pam_appl.h>
int pam_chauthtok(pam_handle_t *pamh, const int flags);
```

Description

`pam_chauthtok` is used to change the authentication token for a given user as indicated by the state associated with the handle *`pamh`*.

`flags` is an optional parameter that may be specified by the following value:

PAM_CHANGE_EXPIRED_AUTHTOK

User's authentication token should only be changed if it has expired.

Additionally, the value of *`flags`* may be logically or'd with `PAM_SILENT`.

RETURN VALUE

PAM_SUCCESS

Success.

PAM_AUTHTOK_ERR

A module was unable to obtain the new authentication token.

PAM_AUTHTOK_RECOVER_ERR

A module was unable to obtain the old authentication token.

PAM_AUTHTOK_LOCK_BUSY

One or more modules were unable to change the authentication token since it is currently locked.

PAM_AUTHTOK_DISABLE_AGING

Authentication token aging has been disabled for at least one of the modules.

PAM_PERM_DENIED

Permission denied.

PAM_TRY_AGAIN

Not all modules were in a position to update the authentication token(s). In such a case, none of the user's authentication tokens are updated.

PAM_USER_UNKNOWN

User is not known to the authentication token changing service.

ERRORS

May be translated to text with `pam_strerror`.

pam_close_session

Name

`pam_close_session` — indicate that an authenticated session has ended

Synopsis

```
#include <security/pam_appl.h>
int pam_close_session(pam_handle_t *pamh, int flags);
```

Description

`pam_close_session` is used to indicate that an authenticated session has ended. It is used to inform the module that the user is exiting a session. It should be possible for the PAM library to open a session and close the same session from different applications.

`flags` may have the value `PAM_SILENT` to indicate that no output should be generated as a result of this function call.

Return Value

PAM_SUCCESS

Success.

PAM_SESSION_ERR

One of the required loaded modules was unable to close a session for the user.

Errors

May be translated to text with `pam_strerror`.

pam_end

Name

pam_end — terminate the use of the PAM library

Synopsis

```
#include <security/pam_appl.h>
int pam_end(pam_handle_t *pamh, int pam_status);
```

Description

pam_end terminates use of the PAM library. On success, the contents of **pamh* are no longer valid, and all memory associated with it is invalid.

Normally, *pam_status* is passed the value PAM_SUCCESS, but in the event of an unsuccessful service application, the appropriate PAM error return value should be used.

Return Value

PAM_SUCCESS

Success.

Errors

May be translated to text with pam_strerror.

pam_fail_delay

Name

`pam_fail_delay` — specify delay time to use on authentication error

Synopsis

```
#include <security/pam_appl.h>
int pam_fail_delay(pam_handle_t *pamh, unsigned int micro_sec);
```

Description

`pam_fail_delay` specifies the minimum delay for the PAM library to use when an authentication error occurs. The actual delay can vary by as much at 25%. If this function is called multiple times, the longest time specified by any of the call will be used.

The delay is invoked if an authentication error occurs during the `pam_authenticate` or `pam_chauthtok` function calls.

Independent of the success of `pam_authenticate` or `pam_chauthtok`, the delay time is reset to its default value of 0 when the PAM library returns control to the application from these two functions.

Return Value

PAM_SUCCESS

: Success.

Errors

May be translated to text with `pam_strerror`.

pam_get_item

Name

`pam_get_item` — obtain the value of the indicated item.

Synopsis

```
#include <security/pam_appl.h>
int pam_get_item(const pam_handle_t *pamh, int item_type, const void **item);
```

Description

`pam_get_item` obtains the value of the indicated *`item_type`*. The possible values of *`item_type`* are the same as listed for `pam_set_item`.

On success, *`item`* contains a pointer to the value of the corresponding item. Note that this is a pointer to the actual data and should not be `free`'d or over-written.

Return Value

PAM_SUCCESS

: Success.

PAM_PERM_DENIED

: Application passed a `NULL` pointer for `item`.

PAM_BAD_ITEM

: Application attempted to get an undefined item.

Errors

May be translated to text with `pam_strerror`.

pam_getenvlist

Name

`pam_getenvlist` — returns a pointer to the complete PAM environment.

Synopsis

```
#include <security/pam_appl.h>
char * const *pam_getenvlist(pam_handle_t *pamh);
```

Description

`pam_getenvlist` returns a pointer to the complete PAM environment. This pointer points to an array of pointers to `NUL`-terminated strings and must be terminated by a `NULL` pointer. Each string has the form "name=value".

The PAM library module allocates memory for the returned value and the associated strings. The calling application is responsible for freeing this memory.

Return Value

`pam_getenvlist` returns an array of string pointers containing the PAM environment. On error, `NULL` is returned.

pam_open_session

Name

pam_open_session — used to indicate that an authenticated session has been initiated

Synopsis

```
#include <security/pam_appl.h>
int pam_open_session(pam_handle_t *pamh, int flags);
```

Description

pam_handle_t is used to indicate that an authenticated session has begun. It is used to inform the module that the user is currently in a session. It should be possible for the PAM library to open a session and close the same session from different applications.

flags may have the value PAM_SILENT to indicate that no output be generated as a rsult of this function call.

Return Value

PAM_SUCCESS

Success.

PAM_SESSION_ERR

One of the loaded modules was unable to open a session for the user.

ERRORS

May be translated to text with pam_strerror.

pam_set_item

Name

`pam_set_item` — (re)set the value of an item.

Synopsis

```
#include <security/pam_appl.h>
int pam_set_item(pam_handle_t *pamh, int item_type, const void *item);
```

Description

`pam_set_item` (re)sets the value of one of the following item_types:

PAM_SERVICE

: service name

PAM_USER

: user name

PAM_TTY

: terminal name

: The value for a device file should include the `/dev/` prefix. The value for graphical, X-based, applications should be the `$DISPLAY` variable.

PAM_RHOST

: remote host name

PAM_CONV

: conversation structure

PAM_RUSER

: remote user name

PAM_USER_PROMPT

: string to be used when prompting for a user's name

: The default value for this string is `Please enter username: `.

For all *`item_types`* other than `PAM_CONV`, *`item`* is a pointer to a `NULL`-terminated character string. In the case of `PAM_CONV`, *`item`* points to an initialized pam_conv structure.

Return Value

PAM_SUCCESS

: Success.

PAM_PERM_DENIED

: An attempt was made to replace the conversation structure with a `NULL` value.

PAM_BUF_ERR

: Function ran out of memory making a copy of the item.

PAM_BAD_ITEM

: Application attempted to set an undefined item.

Errors

May be translated to text with `pam_strerror`.

pam_setcred

Name

`pam_setcred` — set the module-specific credentials of the user

Synopsis

```
#include <security/pam_appl.h>
extern int pam_setcred(pam_handle_t *pamh, int flags);
```

Description

`pam_setcred` sets the module-specific credentials of the user. It is usually called after the user has been authenticated, after the account management function has been called and after a session has been opened for the user.

`flags` maybe specified from among the following values:

PAM_ESTABLISH_CRED

: set credentials for the authentication service

PAM_DELETE_CRED

: delete credentials associated with the authentication service

PAM_REINITIALIZE_CRED

: reinitialize the user credentials

PAM_REFRESH_CRED

: extend lifetime of the user credentials

Additionally, the value of *`flags`* may be logically or'd with `PAM_SILENT`.

Return Value

PAM_SUCCESS

: Success.

PAM_CRED_UNAVAIL

: Module cannot retrieve the user's credentials.

PAM_CRED_EXPIRED

: User's credentials have expired.

PAM_USER_UNKNOWN

: User is not known to an authentication module.

PAM_CRED_ERR

Module was unable to set the credentials of the user.

Errors

May be translated to text with `pam_strerror`.

pam_start

Name

`pam_start` — initialize the PAM library

Synopsis

```
#include <security/pam_appl.h>
int pam_start(const char *service_name, const char *user, const (struct pam_conv
*pam_conversation), pam_handle_t **pamh);
```

Description

`pam_start` is used to initialize the PAM library. It must be called prior to any other usage of the PAM library. On success, *`*pamh`* becomes a handle that provides continuity for successive calls to the PAM library. `pam_start` expects arguments as follows: the *`service_name`* of the program, the *`username`* of the individual to be authenticated, a pointer to an application-supplied pam_conv structure, and a pointer to a *`pam_handle_t`* pointer.

An application must provide the *conversation function* used for direct communication between a loaded module and the application. The application also typically provides a means for the module to prompt the user for a password, etc.

The structure, pam_conv, is defined to be,

```
  struct pam_conv {
              int (*conv) (int num_msg,
                           const struct pam_message * *msg,
                           struct pam_response * *resp,
                           void *appdata_ptr);
              void *appdata_ptr;
```

```
};
```

It is initialized by the application before it is passed to the library. The contents of this structure are attached to the **pamh* handle. The point of this argument is to provide a mechanism for any loaded module to interact directly with the application program; this is why it is called a conversation structure.

When a module calls the referenced `conv` function, *`appdata_ptr`* is set to the second element of this structure.

The other arguments of a call to `conv` concern the information exchanged by module and application. *`num_msg`* holds the length of the array of pointers passed via *`msg`*. On success, the pointer *`resp`* points to an array of *`num_msg`* pam_response structures, holding the application-supplied text. Note that *`resp`* is a struct pam_response array and not an array of pointers.

Return Value

PAM_SUCCESS

: Success.

PAM_BUF_ERR

: Memory allocation error.

PAM_ABORT

: Internal failure.

ERRORS

May be translated to text with `pam_strerror`.

pam_strerror

Name

`pam_strerror` — returns a string describing the PAM error

Synopsis

```
#include <security/pam_appl.h>
const char * pam_strerror(pam_handle_t *pamh, int errnum);
```

Description

`pam_strerror` returns a string describing the PAM error associated with *`errnum`*.

Return Value

On success, this function returns a description of the indicated error. The application should not free or modify this string. This returned string will not be translated.

Notes

1. Future versions of this specification might define additional service names.

II. Utility Libraries

Chapter 2. utility Libraries

An LSB-conforming implementation shall 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 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

adler32 [1]	deflateInit_ [1]	gzerror [1]	gzread [1]	inflateInit2_ [1]
compress [1]	deflateParams [1]	gzflush [1]	gzrewind [1]	inflateInit_ [1]
compress2 [1]	deflateReset [1]	gzgetc [1]	gzseek [1]	inflateReset [1]
crc32 [1]	deflateSetDictionary [1]	gzgets [1]	gzsetparams [1]	inflateSetDictionary [1]
deflate [1]	get_crc_table [1]	gzopen [1]	gztell [1]	inflateSync [1]
deflateCopy [1]	gzclose [1]	gzprintf [1]	gzwrite [1]	inflateSyncPoint [1]
deflateEnd [1]	gzdopen [1]	gzputc [1]	inflate [1]	uncompress [1]
deflateInit2_ [1]	gzeof [1]	gzputs [1]	inflateEnd [1]	zError [1]

Referenced Specification(s)

[1]. zlib 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;
}
 ;
typedef Byte Bytef;
typedef uInt uIntf;

typedef struct z_stream_s
{
  Bytef *next_in;
  uInt avail_in;
```

```
    uLong total_in;
    Bytef *next_out;
    uInt avail_out;
    uLong total_out;
    char *msg;
    struct internal_state *state;
    alloc_func zalloc;
    free_func zfree;
    voidpf opaque;
    int data_type;
    uLong adler;
    uLong reserved;
  }
  z_stream;

  typedef z_stream *z_streamp;
  typedef voidp gzFile;
  #define Z_NO_FLUSH      0
  #define Z_PARTIAL_FLUSH 1
  #define Z_SYNC_FLUSH    2
  #define Z_FULL_FLUSH    3
  #define Z_FINISH        4

  #define Z_ERRNO (-1)
  #define Z_STREAM_ERROR  (-2)
  #define Z_DATA_ERROR    (-3)
  #define Z_MEM_ERROR     (-4)
  #define Z_BUF_ERROR     (-5)
  #define Z_OK    0
  #define Z_STREAM_END    1
  #define Z_NEED_DICT     2

  #define Z_DEFAULT_COMPRESSION   (-1)
  #define Z_NO_COMPRESSION        0
  #define Z_BEST_SPEED    1
  #define Z_BEST_COMPRESSION      9

  #define Z_DEFAULT_STRATEGY      0
  #define Z_FILTERED      1
  #define Z_HUFFMAN_ONLY  2

  #define Z_BINARY        0
  #define Z_ASCII 1
  #define Z_UNKNOWN       2

  #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:

X/Open Curses

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 [1]	has_ic [1]	mvwaddchnstr [1]	scr_init [1]	vwscanw [1]
addchnstr [1]	has_il [1]	mvwaddchstr [1]	scr_restore [1]	waddch [1]
addchstr [1]	hline [1]	mvwaddnstr [1]	scr_set [1]	waddchnstr [1]
addnstr [1]	idcok [1]	mvwaddstr [1]	scrl [1]	waddchstr [1]
addstr [1]	idlok [1]	mvwchgat [1]	scroll [1]	waddnstr [1]
attr_get [1]	immedok [1]	mvwdelch [1]	scrollok [1]	waddstr [1]
attr_off [1]	inch [1]	mvwgetch [1]	set_curterm [1]	wattr_get [1]
attr_on [1]	inchnstr [1]	mvwgetnstr [1]	set_term [1]	wattr_off [1]
attr_set [1]	inchstr [1]	mvwgetstr [1]	setscrreg [1]	wattr_on [1]
attroff [1]	init_color [1]	mvwhline [1]	setupterm [1]	wattr_set [1]
attron [1]	init_pair [1]	mvwin [1]	slk_attr_set [1]	wattroff [1]
attrset [1]	initscr [1]	mvwinch [1]	slk_attroff [1]	wattron [1]
baudrate [1]	innstr [1]	mvwinchnstr [1]	slk_attron [1]	wattrset [1]
beep [1]	insch [1]	mvwinchstr [1]	slk_attrset [1]	wbkgd [1]
bkgd [1]	insdelln [1]	mvwinnstr [1]	slk_clear [1]	wbkgdset [1]
bkgdset [1]	insertln [1]	mvwinsch [1]	slk_color [1]	wborder [1]
border [1]	insnstr [1]	mvwinsnstr [1]	slk_init [1]	wchgat [1]
box [1]	insstr [1]	mvwinsstr [1]	slk_label [1]	wclear [1]
can_change_color [1]	instr [1]	mvwinstr [1]	slk_noutrefresh [1]	wclrtobot [1]

cbreak [1]	intrflush [1]	mvwprintw [1]	slk_refresh [1]	wclrtoeol [1]
chgat [1]	is_linetouched [1]	mvwscanw [1]	slk_restore [1]	wcolor_set [1]
clear [1]	is_wintouched [1]	mvwvline [1]	slk_set [1]	wcursyncup [1]
clearok [1]	isendwin [1]	napms [1]	slk_touch [1]	wdelch [1]
clrtobot [1]	keyname [1]	newpad [1]	standend [1]	wdeleteln [1]
clrtoeol [1]	keypad [1]	newterm [1]	standout [1]	wechochar [1]
color_content [1]	killchar [1]	newwin [1]	start_color [1]	werase [1]
color_set [1]	leaveok [1]	nl [1]	subpad [1]	wgetch [1]
copywin [1]	longname [1]	nocbreak [1]	subwin [1]	wgetnstr [1]
curs_set [1]	meta [1]	nodelay [1]	syncok [1]	wgetstr [1]
def_prog_mode [1]	move [1]	noecho [1]	termattrs [1]	whline [1]
def_shell_mode [1]	mvaddch [1]	nonl [1]	termname [1]	winch [1]
del_curterm [1]	mvaddchnstr [1]	noqiflush [1]	tgetent [1]	winchnstr [1]
delay_output [1]	mvaddchstr [1]	noraw [1]	tgetflag [1]	winchstr [1]
delch [1]	mvaddnstr [1]	notimeout [1]	tgetnum [1]	winnstr [1]
deleteln [1]	mvaddstr [1]	overlay [1]	tgetstr [1]	winsch [1]
delscreen [1]	mvchgat [1]	overwrite [1]	tgoto [1]	winsdelln [1]
delwin [1]	mvcur [1]	pair_content [1]	tigetflag [1]	winsertln [1]
derwin [1]	mvdelch [1]	pechochar [1]	tigetnum [1]	winsnstr [1]
doupdate [1]	mvderwin [1]	pnoutrefresh [1]	tigetstr [1]	winsstr [1]
dupwin [1]	mvgetch [1]	prefresh [1]	timeout [1]	winstr [1]
echo [1]	mvgetnstr [1]	printw [1]	touchline [1]	wmove [1]
echochar [1]	mvgetstr [1]	putp [1]	touchwin [1]	wnoutrefresh [1]
endwin [1]	mvhline [1]	putwin [1]	tparm [1]	wprintw [1]
erase [1]	mvinch [1]	qiflush [1]	tputs [1]	wredrawln [1]
erasechar [1]	mvinchnstr [1]	raw [1]	typeahead [1]	wrefresh [1]
filter [1]	mvinchstr [1]	redrawwin [1]	unctrl [1]	wscanw [1]
flash [1]	mvinnstr [1]	refresh [1]	ungetch [1]	wscrl [1]
flushinp [1]	mvinsch [1]	reset_prog_mode [1]	untouchwin [1]	wsetscrreg [1]
getbkgd [1]	mvinsnstr [1]	reset_shell_mode	use_env [1]	wstandend [1]

		[1]		
getch [1]	mvinsstr [1]	resetty [1]	vidattr [1]	wstandout [1]
getnstr [1]	mvinstr [1]	restartterm [1]	vidputs [1]	wsyncdown [1]
getstr [1]	mvprintw [1]	ripoffline [1]	vline [1]	wsyncup [1]
getwin [1]	mvscanw [1]	savetty [1]	vw_printw [1]	wtimeout [1]
halfdelay [1]	mvvline [1]	scanw [1]	vw_scanw [1]	wtouchln [1]
has_colors [1]	mvwaddch [1]	scr_dump [1]	vwprintw [1]	wvline [1]

Referenced Specification(s)

[1]. X/Open Curses

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 [1]	COLS [1]	acs_map [1]	curscr [1]	
COLOR_PAIRS [1]	LINES [1]	cur_term [1]	stdscr [1]	

Referenced Specification(s)

[1]. X/Open Curses

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

```
#define ERR     (-1)
#define OK      (0)
#define ACS_RARROW      (acs_map['+'])
#define ACS_LARROW      (acs_map[','])
#define ACS_UARROW      (acs_map['-'])
#define ACS_DARROW      (acs_map['.'])
#define ACS_BLOCK       (acs_map['0'])
#define ACS_CKBOARD     (acs_map['a'])
#define ACS_DEGREE      (acs_map['f'])
```

```
142   #define ACS_PLMINUS     (acs_map['g'])
143   #define ACS_BOARD       (acs_map['h'])
144   #define ACS_LANTERN     (acs_map['i'])
145   #define ACS_LRCORNER    (acs_map['j'])
146   #define ACS_URCORNER    (acs_map['k'])
147   #define ACS_ULCORNER    (acs_map['l'])
148   #define ACS_LLCORNER    (acs_map['m'])
149   #define ACS_PLUS        (acs_map['n'])
150   #define ACS_S1  (acs_map['o'])
151   #define ACS_HLINE       (acs_map['q'])
152   #define ACS_S9  (acs_map['s'])
153   #define ACS_LTEE        (acs_map['t'])
154   #define ACS_RTEE        (acs_map['u'])
155   #define ACS_BTEE        (acs_map['v'])
156   #define ACS_TTEE        (acs_map['w'])
157   #define ACS_VLINE       (acs_map['x'])
158   #define ACS_DIAMOND     (acs_map['`'])
159   #define ACS_BULLET      (acs_map['~'])
160   #define getmaxyx(win,y,x)
161   (y=(win)?((win)->_maxy+1):ERR,x=(win)?((win)->_maxx+1):ERR)
162   #define getbegyx(win,y,x)       (y=(win)?(win)->_begy:ERR,x=(win)?(win)->_begx:ERR)
163   #define getyx(win,y,x)  (y=(win)?(win)->_cury:ERR,x=(win)?(win)->_curx:ERR)
164   #define getparyx(win,y,x)       (y=(win)?(win)->_pary:ERR,x=(win)?(win)->_parx:ERR)
165
166   #define WA_ALTCHARSET   A_ALTCHARSET
167   #define WA_ATTRIBUTES   A_ATTRIBUTES
168   #define WA_BLINK        A_BLINK
169   #define WA_BOLD A_BOLD
170   #define WA_DIM  A_DIM
171   #define WA_HORIZONTAL   A_HORIZONTAL
172   #define WA_INVIS        A_INVIS
173   #define WA_LEFT A_LEFT
174   #define WA_LOW  A_LOW
175   #define WA_NORMAL       A_NORMAL
176   #define WA_PROTECT      A_PROTECT
177   #define WA_REVERSE      A_REVERSE
178   #define WA_RIGHT        A_RIGHT
179   #define WA_STANDOUT     A_STANDOUT
180   #define WA_TOP  A_TOP
181   #define WA_UNDERLINE    A_UNDERLINE
182   #define WA_VERTICAL     A_VERTICAL
183   #define A_REVERSE       NCURSES_BITS(1UL,10)
184
185   #define COLOR_BLACK     0
186   #define COLOR_RED       1
187   #define COLOR_GREEN     2
188   #define COLOR_YELLOW    3
189   #define COLOR_BLUE      4
190   #define COLOR_MAGENTA   5
191   #define COLOR_CYAN      6
192   #define COLOR_WHITE     7
193
194   #define _SUBWIN 0x01
```

```
195   #define _ENDLINE        0x02
196   #define _FULLWIN        0x04
197   #define _ISPAD  0x10
198   #define _HASMOVED       0x20
199
200   typedef unsigned char bool;
201
202   typedef unsigned long chtype;
203   typedef struct screen SCREEN;
204   typedef struct _win_st WINDOW;
205   typedef chtype attr_t;
206   typedef struct
207   {
208     attr_t attr;
209     wchar_t chars[5];
210   }
211   cchar_t;
212   struct pdat
213   {
214     short _pad_y;
215     short _pad_x;
216     short _pad_top;
217     short _pad_left;
218     short _pad_bottom;
219     short _pad_right;
220   }
221    ;
222
223   struct _win_st
224   {
225     short _cury;
226     short _curx;
227     short _maxy;
228     short _maxx;
229     short _begy;
230     short _begx;
231     short _flags;
232     attr_t _attrs;
233     chtype _bkgd;
234     bool _notimeout;
235     bool _clear;
236     bool _leaveok;
237     bool _scroll;
238     bool _idlok;
239     bool _idcok;
240     bool _immed;
241     bool _sync;
242     bool _use_keypad;
243     int _delay;
244     struct ldat *_line;
245     short _regtop;
246     short _regbottom;
247     int _parx;
```

```
    int _pary;
    WINDOW *_parent;
    struct pdat _pad;
    short _yoffset;
    cchar_t _bkgrnd;
  }
   ;
  #define KEY_CODE_YES    0400
  #define KEY_BREAK       0401
  #define KEY_MIN 0401
  #define KEY_DOWN        0402
  #define KEY_UP  0403
  #define KEY_LEFT        0404
  #define KEY_RIGHT       0405
  #define KEY_HOME        0406
  #define KEY_BACKSPACE   0407
  #define KEY_F0  0410
  #define KEY_DL  0510
  #define KEY_IL  0511
  #define KEY_DC  0512
  #define KEY_IC  0513
  #define KEY_EIC 0514
  #define KEY_CLEAR       0515
  #define KEY_EOS 0516
  #define KEY_EOL 0517
  #define KEY_SF  0520
  #define KEY_SR  0521
  #define KEY_NPAGE       0522
  #define KEY_PPAGE       0523
  #define KEY_STAB        0524
  #define KEY_CTAB        0525
  #define KEY_CATAB       0526
  #define KEY_ENTER       0527
  #define KEY_SRESET      0530
  #define KEY_RESET       0531
  #define KEY_PRINT       0532
  #define KEY_LL  0533
  #define KEY_A1  0534
  #define KEY_A3  0535
  #define KEY_B2  0536
  #define KEY_C1  0537
  #define KEY_C3  0540
  #define KEY_BTAB        0541
  #define KEY_BEG 0542
  #define KEY_CANCEL      0543
  #define KEY_CLOSE       0544
  #define KEY_COMMAND     0545
  #define KEY_COPY        0546
  #define KEY_CREATE      0547
  #define KEY_END 0550
  #define KEY_EXIT        0551
  #define KEY_FIND        0552
  #define KEY_HELP        0553
```

```
301   #define KEY_MARK        0554
302   #define KEY_MESSAGE     0555
303   #define KEY_MOVE        0556
304   #define KEY_NEXT        0557
305   #define KEY_OPEN        0560
306   #define KEY_OPTIONS     0561
307   #define KEY_PREVIOUS    0562
308   #define KEY_REDO        0563
309   #define KEY_REFERENCE   0564
310   #define KEY_REFRESH     0565
311   #define KEY_REPLACE     0566
312   #define KEY_RESTART     0567
313   #define KEY_RESUME      0570
314   #define KEY_SAVE        0571
315   #define KEY_SBEG        0572
316   #define KEY_SCANCEL     0573
317   #define KEY_SCOMMAND    0574
318   #define KEY_SCOPY       0575
319   #define KEY_SCREATE     0576
320   #define KEY_SDC 0577
321   #define KEY_SDL 0600
322   #define KEY_SELECT      0601
323   #define KEY_SEND        0602
324   #define KEY_SEOL        0603
325   #define KEY_SEXIT       0604
326   #define KEY_SFIND       0605
327   #define KEY_SHELP       0606
328   #define KEY_SHOME       0607
329   #define KEY_SIC 0610
330   #define KEY_SLEFT       0611
331   #define KEY_SMESSAGE    0612
332   #define KEY_SMOVE       0613
333   #define KEY_SNEXT       0614
334   #define KEY_SOPTIONS    0615
335   #define KEY_SPREVIOUS   0616
336   #define KEY_SPRINT      0617
337   #define KEY_SREDO       0620
338   #define KEY_SREPLACE    0621
339   #define KEY_SRIGHT      0622
340   #define KEY_SRSUME      0623
341   #define KEY_SSAVE       0624
342   #define KEY_SSUSPEND    0625
343   #define KEY_SUNDO       0626
344   #define KEY_SUSPEND     0627
345   #define KEY_UNDO        0630
346   #define KEY_MOUSE       0631
347   #define KEY_RESIZE      0632
348   #define KEY_MAX 0777
349
350   #define PAIR_NUMBER(a)  (((a)& A_COLOR)>>8)
351   #define NCURSES_BITS(mask,shift)        ((mask)<<((shift)+8))
352   #define A_CHARTEXT      (NCURSES_BITS(1UL,0)-1UL)
353   #define A_NORMAL        0L
```

```
#define NCURSES_ATTR_SHIFT      8
#define A_COLOR NCURSES_BITS(((1UL)<<8)-1UL,0)
#define A_BLINK NCURSES_BITS(1UL,11)
#define A_DIM   NCURSES_BITS(1UL,12)
#define A_BOLD  NCURSES_BITS(1UL,13)
#define A_ALTCHARSET    NCURSES_BITS(1UL,14)
#define A_INVIS NCURSES_BITS(1UL,15)
#define A_PROTECT       NCURSES_BITS(1UL,16)
#define A_HORIZONTAL    NCURSES_BITS(1UL,17)
#define A_LEFT  NCURSES_BITS(1UL,18)
#define A_LOW   NCURSES_BITS(1UL,19)
#define A_RIGHT NCURSES_BITS(1UL,20)
#define A_TOP   NCURSES_BITS(1UL,21)
#define A_VERTICAL      NCURSES_BITS(1UL,22)
#define A_STANDOUT      NCURSES_BITS(1UL,8)
#define A_UNDERLINE     NCURSES_BITS(1UL,9)
#define COLOR_PAIR(n)   NCURSES_BITS(n,0)
#define A_ATTRIBUTES    NCURSES_BITS(~(1UL-1UL),0)
```

2.5. Interfaces for libutil

Table 2-6 defines the library name and shared object name for the libutil library

Table 2-6. libutil Definition

Library:	libutil
SONAME:	libutil.so.1

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

this specification

2.5.1. Utility Functions

2.5.1.1. Interfaces for Utility Functions

An LSB conforming implementation shall provide the generic functions for Utility Functions specified in Table 2-7, with the full functionality as described in the referenced underlying specification.

Table 2-7. libutil - Utility Functions Function Interfaces

forkpty [1]	login_tty [1]	logwtmp [1]		
login [1]	logout [1]	openpty [1]		

Referenced Specification(s)

[1]. 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` — Create a new process attached to an available pseudo-terminal

Synopsis

```
#include <pty.h>
int forkpty(int * amaster, char * name, struct termios * termp, struct winsize * winp);
```

Description

The `forkpty()` function shall find and open a pseudo-terminal device pair in the same manner as the `openpty()` function. If a pseudo-terminal is available, `forkpty` shall create a new process in the same manner as the `fork()` function, and prepares the new process for login in the same manner as `login_tty()`.

If *`termp`* is not null, it shall refer to a `termios` structure that shall be used to initialize the characteristics of the slave device. If *`winp`* is not null, it shall refer to a `winsize` structure used to initialize the window size of the slave device.

Return Value

On success, the parent process shall return the process id of the child, and the child shall return 0. On error, no new process shall be created, -1 shall be returned, and `errno` shall be set appropriately. On success, the parent process shall receive the file descriptor of the master side of the pseudo-terminal in the location referenced by *`amaster`*, and, if *`name`* is not NULL, the filename of the slave device in *`name`*.

Errors

EAGAIN

: Unable to create a new process.

ENOENT

: There are no available pseudo-terminals.

ENOMEM

: Insufficient memory was available.

login

Name

`login` — login utility function

Synopsis

```
#include <utmp.h>
void login (struct utmp * ut );
```

Description

The `login` function shall update the user accounting databases. The *`ut`* parameter shall reference a utmp structure for all fields except the following:

1. The *`ut_type`* field shall be set to `USER_PROCESS`.
2. The *`ut_pid`* field shall be set to the process identifier for the current process.
3. The *`ut_line`* field shall be set to the name of the controlling terminal device. The name shall be found by examaning the device associated with the standard input, output and error streams in sequence, until one associated with a terminal device is found. If none of these streams refers to a terminal device, the *`ut_line`* field shall be set to `"???"`. If the terminal device is in the `/dev` directory hierarchy, the *`ut_line`* field shall not contain the leading `"/dev/"`, otherwise it shall be set to the final component of the pathname of the device. If the user accounting database imposes a limit on the size of the *`ut_line`* field, it shall truncate the name, but any such limit shall not be smaller than `UT_LINESIZE` (including a terminating null character).

Return Value

None

Errors

None

login_tty

Name

`login_tty` — Prepare a terminal for login

Synopsis

```
#include <utmp.h>
int login_tty (int fdr);
```

Description

The `login_tty()` function shall prepare the terminal device referenced by the file descriptor *`fdr`*. This function shall create a new session, make the terminal the controlling terminal for the current process, and set the standard input, output, and error streams of the current process to the terminal. If *`fdr`* is not the standard input, output or error stream, then `login_tty()` shall close *`fdr`*.

Return Value

On success, `login_tty()` shall return zero; otherwise -1 is returned, and errno shall be set appropriately.

Errors

ENOTTY

`fdr` does not refer to a terminal device.

logout

Name

`logout` — logout utility function

Synopsis

```
#include <utmp.h>
int logout (const char * line );
```

Description

Given the device *`line`*, the `logout` function shall search the user accounting database which is read by `getutent` for an entry with the corresponding line, and with the type of `USER_PROCESS`. If a corresponding entry is located, it shall be updated as follows:

1. The `ut_name` field shall be set to zeroes (`UT_NAMESIZE` NUL bytes).
2. The `ut_host` field shall be set to zeroes (`UT_HOSTSIZE` NUL bytes).
3. The `ut_tv` shall be set to the current time of day.
4. The `ut_type` field shall be set to `DEAD_PROCESS`.

Return Value

On success, the `logout()` function shall return non-zero. Zero is returned if there was no entry to remove, or if the utmp file could not be opened or updated.

logwtmp

Name

`logwtmp` — append an entry to the wtmp file

Synopsis

```
#include <utmp.h>
void logwtmp (const char * line , const char * name , const char * host );
```

Description

If the process has permission to update the user accounting databases, the `logwtmp` function shall append a record to the user accounting database that records all logins and logouts. The record to be appended shall be constructed as follows:

1. The `ut_line` field shall be intitialized from *`line`*. If the user accounting database imposes a limit on the size of the *`ut_line`* field, it shall truncate the value, but any such limit shall not be smaller than `UT_LINESIZE` (including a terminating null character).
2. The `ut_name` field shall be intitialized from *`name`*. If the user accounting database imposes a limit on the size of the *`ut_name`* field, it shall truncate the value, but any such limit shall not be smaller than `UT_NAMESIZE` (including a terminating null character).
3. The `ut_host` field shall be intitialized from *`host`*. If the user accounting database imposes a limit on the size of the *`ut_host`* field, it shall truncate the value, but any such limit shall not be smaller than `UT_HOSTSIZE` (including a terminating null character).
4. If the *`name`* parameter does not refer to an empty string (i.e. `""`), the `ut_type` field shall be set to `USER_PROCESS`; otherwise the `ut_type` fieldshall be set to `DEAD_PROCESS`.
5. The `ut_id` field shall be set to the process identifier for the current process.
6. The `ut_tv` field shall be set to the current time of day.

> If a process does not have write access to the the user accounting database, the `logwtmp` function will not update it. Since the function does not return any value, an application has no way of knowing whether it succeeded or failed.

Return Value

None.

openpty

Name

`openpty` — find and open an available pseudo-terminal

Synopsis

```
#include <pty.h>
int openpty(int *amaster, int *aslave, char *name, struct termios *termp, struct winsize
*winp);
```

Description

The `openpty()` function shall find an available pseudo-terminal and return file descriptors for the master and slave devices in the locations referenced by *`amaster`* and *`aslave`* respectively. If *`name`* is not NULL, the filename of the slave shall be placed in the user supplied buffer referenced by *`name`*. If *`termp`* is not NULL, it shall point to a `termios` structure used to initialize the terminal parameters of the slave pseudo-terminal device. If *`winp`* is not NULL, it shall point to a `winsize` structure used to initialize the window size parameters of the slave pseudo-terminal device.

Return Value

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

Errors

ENOENT

: There are no available pseudo-terminals.

III. Commands and Utilities

Chapter 3. Commands and Utilities

3.1. Commands and Utilities

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 applictions 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.

this specification
ISO POSIX (2003)

Table 3-1. Commands and Utilities

[[1]	ar [2]	at [2]	awk [2]	basename [1]
batch [2]	bc [2]	cat [1]	chfn [2]	chgrp [2]
chmod [1]	chown [2]	chsh [2]	cksum [1]	cmp [1]
col [2]	comm [1]	cp [1]	cpio [2]	crontab [2]
csplit [1]	cut [2]	date [1]	dd [1]	df [2]
diff [1]	dirname [1]	dmesg [2]	du [2]	echo [2]
egrep [2]	env [1]	expand [1]	expr [1]	false [1]
fgrep [2]	file [2]	find [2]	fold [1]	fuser [2]
gencat [1]	getconf [1]	gettext [2]	grep [2]	groupadd [2]
groupdel [2]	groupmod [2]	groups [2]	gunzip [2]	gzip [2]
head [1]	hostname [2]	iconv [1]	id [1]	install [2]
install_initd [2]	ipcrm [2]	ipcs [2]	join [1]	kill [1]
killall [2]	ln [1]	locale [1]	localedef [1]	logname [1]
lpr [2]	ls [2]	lsb_release [2]	m4 [2]	make [1]
man [1]	md5sum [2]	mkdir [1]	mkfifo [1]	mknod [2]

mktemp [2]	more [2]	mount [2]	msgfmt [2]	mv [1]
newgrp [2]	nice [1]	nl [1]	nohup [1]	od [2]
passwd [2]	paste [1]	patch [2]	pathchk [1]	pidof [2]
pr [1]	printf [1]	ps [1]	pwd [1]	remove_initd [2]
renice [2]	rm [1]	rmdir [1]	sed [2]	sendmail [2]
sh [1]	shutdown [2]	sleep [1]	sort [1]	split [1]
strip [1]	stty [1]	su [2]	sync [2]	tail [1]
tar [2]	tee [1]	test [1]	time [1]	touch [1]
tr [1]	true [1]	tsort [1]	tty [1]	umount [2]
uname [1]	unexpand [1]	uniq [1]	useradd [2]	userdel [2]
usermod [2]	wc [1]	xargs [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

ar — create and maintain library archives (LSB DEPRECATED)

Description

ar is deprecated from the LSB and is expected to disappear from a future version of the LSB.

Rationale

The LSB generally does not include software development utilities nor does it specify .o and .a file formats.

ar is as specified in ISO POSIX (2003) but with differences as listed below.

Differences

-T

-C

need not be accepted.

-l

has unspecified behavior.

-q

has unspecified behavior; using -r is suggested.

at

Name

`at` — examine or delete jobs for later execution

Description

at is as specified in ISO POSIX (2003) but with differences as listed below.

Differences

-d

is functionally equivalent to the -r option specified in ISO POSIX (2003).

-r

need not be supported, but the '-d' option is equivalent.

-t time

need not be supported.

Files

The files at.allow and at.deny reside in /etc rather than /usr/lib/cron.

awk

Name

`awk` — pattern scanning and processing language

Description

awk is as specified in ISO POSIX (2003) but with differences as listed below.

Differences

Certain aspects of internationalized regular expressions are optional; see Internationalization and Regular Expressions>.

batch

Name

`batch` — schedule commands to be executed in a batch queue

Description

The specification for **batch** is as specified in ISO POSIX (2003), but with the following differences as listed below.

Files

The files `at.allow` and `at.deny` reside in `/etc` rather than `/usr/lib/cron`.

bc

Name

`bc` — An arbitrary precision calculator language

Description

bc is as specified in ISO POSIX (2003) but with differences as listed below.

Differences

The bc language may be extended in an implementation defined manner. If an implementation supports extensions, it shall also support the additional options:

-s|--standard

: processes exactly the POSIX bc language.

-w|--warn

: gives warnings for extensions to POSIX bc.

chfn

Name

`chfn` — change user name and information

Synopsis

chfn [-f *full_name*] [-h *home_phone*] [user]

Description

chfn shall update the user database. An unprivileged user may only change the fields for their own account, a user with appropriate privileges may change the fields for any account.

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.

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.

chgrp

Name

`chgrp` — change file group

Description

chgrp is as specified in ISO POSIX (2003) but with differences as listed below.

Differences

The -L, -H, and -P options need not be supported.

chown

Name

`chown` — change file owner and group

Description

chown is as specified in ISO POSIX (2003) but with differences as listed below.

Differences

The -L, -H, and -P options need not be supported.

chsh

Name

`chsh` — change login shell

Synopsis

chsh [-s *`login_shell`*] [*`user`*]

Description

chsh changes the user login shell. This determines the name of the user's initial login command. An unprivileged user may only change the login shell for their own account, a user with appropriate privilege may change the login shell for any account specified by *`user`*.

Unless the user has appropriate privilege, the initial login command name shall be one of those listed in `/etc/shells`. The *`login_shell`* shall be the absolute path (i.e. it must start with '`/`') to an executable file. Accounts which are restricted (in an implementation-defined manner) may not change their login shell.

If the `-s` option is not selected, **chsh** operates in an interactive mode. The prompts and expected input in this mode are unspecified.

Standard Options

`-s login_shell`

sets the login shell.

col

Name

`col` — filter reverse line feeds from input

Description

col is as specified in the SUSv2 with the difference that the **-p** option has unspecified behavior.

Although **col** is shown as legacy in SUSv2, Version 2, it is not (yet) deprecated in the LSB.

cpio

Name

`cpio` — copy file archives in and out

Description

cpio is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

Some elements of the Pattern Matching Notation are optional; see Internationalization and Pattern Matching Notation.

crontab

Name

`crontab` — maintain crontab files for individual users

Synopsis

crontab [-u user] file
crontab [-u user] {-l | -r | -e}

Description

crontab is as specified in ISO POSIX (2003), but with differences as listed below.

Files

The files `cron.allow` and `cron.deny` reside in `/etc` rather than `/usr/lib/cron`.

cut

Name

`cut` — split a file into sections determined by context lines

Description

cut is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

-n

has unspecified behavior.

df

Name

`df` — report filesystem disk space usage

Description

df is as specified in ISO POSIX (2003), but with the following differences.

If the *-k* option is not specified, disk space is shown in unspecified units. Applications should specify *-k*.

If an argument is the absolute file name of a disk device node containing a mounted filesystem, df shows the space available on that filesystem rather than on the filesystem containing the device node (which is always the root filesystem).

dmesg

Name

`dmesg` — print or control the system message buffer

Synopsis

dmesg [-c | -n *level* | -s *bufsize*]

Description

dmesg examines or controls the system message buffer. Only a user with appropriate privileges may modify the system message buffer parameters or contents.

Standard Options

`-c`

If the user has appropriate privilege, clears the system message buffer contents after printing.

`-n level`

If the user has appropriate privilege, sets the level at which logging of messages is done to the console.

`-s bufsize`

uses a buffer of *bufsize* to query the system message buffer. This is 16392 by default (this matches the default kernel syslog buffer size since 2.1.113). 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

`du` — estimate file space usage

Description

du is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

If the `-k` option is not specified, disk space is shown in unspecified units. Applications should specify `-k`.

echo

Name

`echo` — display a line of text

Synopsis

echo [STRING...]

Description

The **echo** command is as specified in ISO POSIX (2003), but with the following differences.

Unlike the behavior specified in ISO POSIX (2003), whether **echo** supports options is implementation defined. The behavior of **echo** if any arguments contain backslashes is also implementation defined. Conforming applications shall not run **echo** with a first argument starting with a hyphen, or with any arguments containing backslashes; they shall use **printf** in those cases.

> The behavior specified here is similar to that specified by ISO POSIX (2003) without the XSI option. However, the LSB forbids all options and the latter forbids only *-n*.

egrep

Name

`egrep` — search a file with an ERE pattern

Description

egrep is equivalent to grep -E. For further details, see the specification for **grep**.

fgrep

Name

`fgrep` — search a file with a fixed pattern

Description

fgrep is equivalent to grep -F. For further details, see the specification for **grep**.

file

Name

`file` — determine file type

Description

file is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

The *-M*, *-h*, *-d*, and *-i* options need not be supported.

find

Name

`find` — search for files in a directory hierarchy

Description

find is as specified in ISO POSIX (2003), but with additional options as specified below.

Differences

Some elements of the Pattern Matching Notation are optional; see Internationalization and Pattern Matching Notation.

fuser

Name

`fuser` — identify processes using files or sockets

Description

fuser is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

-c

has unspecified behavior.

-f

has unspecified behavior.

gettext

Name

`gettext` — retrieve text string from message catalog

Synopsis

gettext [options] [textdomain] msgid
gettext -s [options] msgid...

Description

The **gettext** utility retrieves a translated text string corresponding to string *`msgid`* from a message object generated with **msgfmt** utility.

The message object name is derived from the optional argument *`textdomain`* if present, otherwise from the `TEXTDOMAIN` environment variable. If no domain is specified, or if a corresponding string cannot be found, **gettext** prints *`msgid`*.

Ordinarily **gettext** looks for its message object in *`dirname`*`/`*`lang`*`/LC_MESSAGES` where *`dirname`* is the implementation-defined default directory and *`lang`* is the locale name. If present, the `TEXTDOMAINDIR` environment variable replaces the *`dirname`*.

This utility interprets C escape sequences such as `\t` for tab. Use `\\` to print a backslash. To produce a message on a line of its own, either put a `\n` at the end of *`msgid,`* or use this command in conjunction with the **printf** utility.

When used with the *`-s`* option the **gettext** utility behaves like the **echo** utility, except that the message corresponding to *`msgid`* in the selected catalog provides the arguments.

Options

`-d domainname`
`--domain=domainname`

PARAMETER translated messages from domainname.

`-e`

Enable expansion of some escape sequences.

`-n`

Suppress trailing newline.

Operands

The following operands are supported:

`textdomain`

A domain name used to retrieve the messages.

msgid

A key to retrieve the localized message.

Environment Variables

LANGUAGE

Specifies one or more locale names.

LANG

Specifies locale name.

LC_MESSAGES

Specifies messaging locale, and if present overrides LANG for messages.

TEXTDOMAIN

Specifies the text domain name, which is identical to the message object filename without .mo suffix.

TEXTDOMAINDIR

Specifies the pathname to the message catalog, and if present replaces the implementation-defined default directory.

Exit Status

The following exit values are returned:

0

Successful completion.

>0

An error occurred.

grep

Name

grep — print lines matching a pattern

Description

grep is as specified in ISO POSIX (2003), but with differences as listed below.

LSB Differences

Some elements of the Pattern Matching Notation are optional; see Internationalization and Pattern Matching Notation.

groupadd

Name

`groupadd` — create a new group

Synopsis

groupadd [-g gid [-o]] group

Description

If the caller has appropriate privilege, the **groupadd** command shall create a new group named *`group`*. The group name shall be unique in the group database. If no *`gid`* is specified, **groupadd** shall create the new group with a unique group ID.

Options

`-g gid [-o]`

The new group shall have group ID *`gid`*. If the `-o` option is not used, no other group shall have this group ID. The value of *`gid`*shall be non-negative.

groupdel

Name

`groupdel` — delete a group

Synopsis

groupdel group

Description

If the caller has sufficient privilege, the **groupdel** command shall modify the system group database, deleting the group named *`group`*. If the group named *`group`* does not exist, **groupdel** shall issue a diagnostic message and exit with a non-zero exit status.

groupmod

Name

`groupmod` — modify a group

Synopsis

groupmod [-g *`gid`* [-o]] [-n *`group_name`*] group

Description

If the caller has appropriate privilege, the **groupmod** command shall modify the entry in the system group database corresponding to a group named *`group`*.

Options

`-g gid [-o]`

Modify the group's group ID, setting it to *`gid`*. If the *`-o`* option is not used, no other group shall have this group ID. The value of *`gid`*shall be non-negative.

> Only the group ID in the database is altered; any files with group ownership set to the original group ID are unchanged by this modification.

`-n group_name`

changes the name of the group from *`group`* to *`group_name`*.

groups

Name

`groups` — display a group

Synopsis

groups [user]

Description

The **groups** command shall behave as **id -Gn *`[user]`***, as specified in ISO POSIX (2003). The optional *`user`* parameter will display the groups for the named user.

gunzip

Name

`gunzip` — uncompress files

Description

gunzip is equivalent to **gzip -d**. See the specification for **gzip** for further details.

gzip

Name

`gzip` — compress or expand files

Synopsis

gzip [-acdfhlLnNrtvV19] [-S suffix] [name...]

Description

The **gzip** command shall attempt to reduce the size of the named files. Whenever possible, each file is replaced by one with the extension `.gz`, while keeping the same ownership modes, access and modification times. If no files are specified, or if a file name is `-`, the standard input is compressed to the standard output. **gzip** shall only attempt to compress regular files. In particular, it will ignore symbolic links.

When compressing, gzip uses the deflate algorithm specified in RFC 1951: DEFLATE Compressed Data Format Specification and stores the result in a file using the gzip file format specified in RFC 1952: GZIP File Format Specification.

Options

-a, --ascii

does nothing on LSB conforming systems.

This option may be deprecated in a future verion of this specification.

-c, --stdout, --to-stdout

writes output on standard output, leaving the original files unchanged. If there are several input files, the output consists of a sequence of independently compressed members. To obtain better compression, concatenate all input files before compressing them.

-d, --decompress, --uncompress

the name operands are compressed files, and **gzip** shall decompress them.

-f, --force

forces compression or decompression even if the file has multiple links or the corresponding file already exists, or if the compressed data is read from or written to a terminal. If the input data is not in a format recognized by **gzip**, and if the option `--stdout` is also given, copy the input data without change to the standard ouput: let **gzip** behave as **cat**. If `-f` is not given, and when not running in the background, **gzip** prompts to verify whether an existing file should be overwritten.

-l, --list

lists the compressed size, uncompressed size, ration and uncompressed name for each compressed file. Gives the uncompressed size as `-1` for files not in **gzip** format. Additionally displays method, crc and timestamp for the uncompress file when used in combination with `--verbose`.

For decompression, **gzip** shall support at least the following compression methods:

- deflate (RFC 1951: DEFLATE Compressed Data Format Specification)
- compress (ISO POSIX (2003))
- lzh (SCO **compress** -**H**)
- pack (Huffman encoding)

The crc shall be given as `ffffffff` for a file not in **gzip** format.

With *`--name`*, the uncompressed name, date and time are those stored within the compress file, if present.

With *`--verbose`*, the size totals and compression ratio for all files is also displayed, unless some sizes are unknown. With *`--quiet`*, the title and totals lines are not displayed.

-L, --license

displays the **gzip** license and quit.

-n, --no-name

does not save the original file name and time stamp by default when compressing. (The original name is always saved if the name had to be truncated.) When decompressing, do not restore the original file name if present (remove only the gzip suffix from the compressed file name) and do not restore the original time stamp if present (copy it from the compressed file). This option is the default when decompressing.

-N, --name

always saves the original file name and time stamp when compressing; this is the default. When decompressing, restore the original file name and time stamp if present. This option is useful on systems which have a limit on file name length or when the time stamp has been lost after a file transfer.

-q, --quiet

suppresses all warnings.

-r, --recursive

travels the directory structure recursively. If any of the file names specified on the command line are directories, **gzip** will descend into the directory and compress all the files it finds there (or decompress them in the case of **gunzip**).

-S .suf, --sufix .suf

uses suffix `.suf` instead of `.gz`.

-t, --test

checks the compressed file integrity.

-v, --verbose

displays the name and percentage reduction for each file compressed or decompressed.

-#, --fast, --best

regulates the speed of compression using the specified digit #, where *-1* or *--fast* indicates the fastest compression method (less compression) and *-9* or *--best* indicates the slowest compression method (best compression). The default compression level is *-6* (that is, biased towards high compression at expense of speed).

LSB Deprecated Options

The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should only use the non-LSB-deprecated behaviors.

-V, --version

displays the version number and compilation options, then quits.

hostname

Name

`hostname` — show or set the system's host name

Synopsis

hostname [name]

Description

hostname is used to either display or, with appropriate privileges, set the current host name of the system. The host name is used by many applications to identify the machine.

When called without any arguments, the program displays the name of the system as returned by the `gethostname` function.

When called with a *name* argument, and the user has appropriate privilege, the command sets the host name.

It is not specified if the hostname displayed will be a fully qualified domain name. Applications requiring a particular format of hostname should check the output and take appropriate action.

install

Name

`install` — copy files and set attributes

Synopsis

install [option...] SOURCE DEST
install [option...] SOURCE... DEST
install [-d | --directory] [option...] DIRECTORY...

Description

In the first two formats, copy *`SOURCE`* to *`DEST`* or multiple *`SOURCE(s)`* to the existing *`DIRECTORY`*, optionally setting permission modes and file ownership. In the third format, each *`DIRECTORY`* and any missing parent directories shall be created.

Standard Options

--backup[=METHOD]

makes a backup of each existing destination file. *`METHOD`* may be one of the following:

- *`none`* or *`off`* never make backups.
- *`numbered`* or *`t`* make numbered backups. A numbered backup has the form `"%s.~%d~", target_name, version_number`. Each backup shall increment the version number by 1.
- *`existing`* or *`nil`* numbered if numbered backups exist, or simple otherwise.
- *`simple`* or *`never`* append a suffix to the name. The default suffix is '~', but can be overriden by setting SIMPLE_BACKUP_SUFFIX in the environment, or via the *`-S`* or *`--suffix`* option.

If no *`METHOD`* is specified, the environment variable VERSION_CONTROL shall be examined for one of the above. Unambiguous abbreviations of *`METHOD`* shall be accepted. If no *`METHOD`* is specified, or if *`METHOD`* is empty, the backup method shall default to `existing`.

If *`METHOD`* is invalid or ambiguous, **install** shall fail and issue a diagnostic message.

-b

is equivalent to *`--backup=existing`*.

-d, --directory

treats all arguments as directory names; creates all components of the specified directories.

-D

creates all leading components of DEST except the last, then copies SOURCE to DEST; useful in the 1st format.

-g GROUP, --group=GROUP

if the user has appropriate privilege, sets group ownership, instead of process' current group. *GROUP* is either a name in the user group database, or a positive integer, which shall be used as a group-id.

-m MODE, --mode=MODE

sets permission mode (specified as in **chmod**), instead of the default `rwxr-xr-x`.

-o OWNER, --owner=OWNER

if the user has appropriate privilege, sets ownership. *OWNER* is either a name in the user login database, or a positive integer, which shall be used as a user-id.

-p, --preserve-timestamps

copies the access and modification times of *SOURCE* files to corresponding destination files.

-s, --strip

strips symbol tables, only for 1st and 2nd formats.

-S SUFFIX, --suffix=SUFFIX

equivalent to `--backup=existing`, except if a simple suffix is required, use *SUFFIX*.

--verbose

prints the name of each directory as it is created.

-v, --verbose

print the name of each file before copying it to `stdout`.

install_initd

Name

`install_initd` — install an init.d file

Synopsis

/usr/lib/lsb/install_initd initd_file

Description

install_initd shall install a system initialization file that has been copied to the `/etc/init.d` location such that this file shall be run at the appropriate point during system initialization. The **install_initrd** command is typically called in the postinstall script of a package. See also Section 8.4.

ipcrm

Name

`ipcrm` — Remove IPC Resources

Synopsis

ipcrm [-q *msgid* | -Q *msgkey* | -s *semid* | -S *semkey* | -m *shmid* | -M *shmkey*]...
ipcrm [shm | msg | msg] id...

Description

If any of the *-q*, *-Q*, *-s*, *-S*, *-m*, *or -M* arguments are given, the **ipcrm** shall behave as described in ISO POSIX (2003).

Otherwise, **ipcrm** shall remove the resource of the specified type identified by *id*.

Future Directions

A future revision of this specification may deprecate the second synopsis form.

Rationale

In its first Linux implementation, **ipcrm** used the second syntax shown in the SYNOPSIS. Functionality present in other implementations of **ipcrm** has since been added, namely the ability to delete resources by key (not just identifier), and to respect the same command line syntax. The previous syntax is still supported for backwards compatibility only.

ipcs

Name

`ipcs` — provide information on ipc facilities

Synopsis

ipcs [-smq] [-tcp]

Description

ipcs provides information on the ipc facilities for which the calling process has read access.

Resource display options

-m

shared memory segments.

-q

message queues.

-s

semaphore arrays.

Output format options

-t

time.

-p

pid.

-c

creator.

Application Usage

In some implementations of ipcs the -a option will print all information available. In other implementations the -a option will print all resource types. Therefore, applications shall not use the -a option.

Some implements of ipcs implement more output formats than are specified here. These options are not consistent between differing implementations of ipcs. Therefore, only the -t -c and -p option flags may be used. At least one of the -t -c and -p options shall be specified.

killall

Name

`killall` — kill processes by name

Synopsis

killall [-egiqvw] [-signal] name...
killall -l
killall -V

Description

killall sends a signal to all processes running any of the specified commands. If no signal name is specified, `SIGTERM` is sent.

Signals can be specified either by name (e.g. `-HUP`) or by number (e.g. `-1`). Signal `0` (check if a process exists) can only be specified by number.

If the command name contains a slash (/), processes executing that particular file will be selected for killing, independent of their name.

killall returns a non-zero return code if no process has been killed for any of the listed commands. If at least one process has been killed for each command, **killall** returns zero.

A **killall** process never kills itself (but may kill other **killall** processes).

Standard Options

-e

: requires an exact match for very long names. If a command name is longer than 15 characters, the full name may be unavailable (i.e. it is swapped out). In this case, **killall** will kill everything that matches within the first 15 characters. With -e, such entries are skipped. **killall** prints a message for each skipped entry if -v is specified in addition to -e.

-g

: kills the process group to which the process belongs. The kill signal is only sent once per group, even if multiple processes belonging to the same process group were found.

-i

: asks interactively for confirmation before killing.

-l

: lists all known signal names.

-q

: does not complain if no processes were killed.

425 -v

426 reports if the signal was successfully sent.

LSB Deprecated Options

427 The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should
428 only use the non-LSB-deprecated behaviors.

429 -V

430 displays version information.

lpr

Name

`lpr` — off line print

Synopsis

lpr [-l] [-p] [-Pprinter] [-h] [-s] [-#copies] [-J name] [-T title] [name .....]

Description

lpr uses a spooling daemon to print the named files when facilities become available. If no names appear, the standard input is assumed.

Standard Options

-l

identifies binary data that is not to be filtered but sent as raw input to printer.

-p

formats with "pr" before sending to printer.

-Pprinter

sends output to the printer named printer instead of the default printer.

-h

suppresses header page.

-s

uses symbolic links.

-#copies

specifies copies as the number of copies to print.

-J name

specifies name as the job name for the header page.

-T title

specifies title as the title used for "pr".

ls

Name

`ls` — list directory contents

Description

ls is as specified in ISO POSIX (2003), but with differences listed below.

Differences

-l

If the file is a character special or block special file, the size of the file shall be replaced with two unsigned numbers in the format `"%u, %u"`, representing the major and minor device numbers associated with the special file.

The LSB does not specify the meaning of the major and minor devices numbers.

-p

in addition to ISO POSIX (2003) behavior of printing a slash for a directory, **ls -p** may display other characters for other file types.

Certain aspects of the pattern matching notation are optional; see Internationalization and Pattern Matching Notation.

lsb_release

Name

`lsb_release` — print distribution specific information

Synopsis

lsb_release [OPTION...]

Description

The **lsb_release** command prints certain LSB (Linux Standard Base) and Distribution information.

If no options are given, the `-v` option is assumed.

Options

-v, --version

displays version of LSB against which distribution is compliant. The version is expressed as a colon seperated list of LSB module descriptions. LSB module descriptions are dash seperated tuples containing the module name, version, and architecture name. The output is a single line of text of the following format:

```
LSB Version:\t<ListAsDescribedAbove>
```

-i, --id

displays string id of distributor. The output is a single line of text of the following format:

```
Distributor ID:\t<DistributorID>
```

-d, --description

displays single line text description of distribution. The output is of the following format:

```
Description:\t<Description>
```

-r, --release

displays release number of distribution. The output is a single line of text of the following format:

```
Release:\t<Release>
```

-c, --codename

displays codename according to distribution release. The output is a single line of text of the following format.

```
Codename:\t<Codename>
```

-a, --all

displays all of the above information.

-s, --short

displays all of the above information in short output format.

-h, --help

displays a human-readable help message.

Examples

The following command will list the LSB Profiles which are currently supported on this platform.

```
example% lsb_release -v
LSB Version: core-2.0-ia32:core-2.0-noarch:graphics-2.0-ia32:graphics-2.0-noarch
```

m4

Name

`m4` — macro processor

Description

m4 is as specified in ISO POSIX (2003), but with extensions as listed below.

Extensions

-P

forces all builtins to be prefixed with `m4_`. For example, `define` becomes `m4_define`.

-I *`directory`*

Add *`directory`* to the end of the search path for includes.

md5sum

Name

`md5sum` — generate or check MD5 message digests

Synopsis

md5sum [-c [file] | file]

Description

For each file, write to standard output a line containing the MD5 message digest of that file, followed by one or more blank characters, followed by the name of the file. The MD5 message digest shall be calculated according to RFC 1321: The MD5 Message-Digest Algorithm and output as 32 hexadecimal digits.

If no file names are specified as operands, read from standard input and use `"-"` as the file name in the output.

Options

-c [file]

checks the MD5 message digest of all files named in *`file`* against the message digest listed in the same file. The actual format of *`file`* is the same as the output of **md5sum**. That is, each line in the file describes a file. If *`file`* is not specified, read message digests from `stdin`.

Exit Status

md5sum shall exit with status 0 if the sum was generated successfully, or, in check mode, if the check matched. Otherwise, **md5sum** shall exit with a non-zero status.

mknod

Name

mknod — make special files

Synopsis

mknod [-m *mode* | --mode=*mode*] name type [major minor]
mknod [--version]

Description

The **mknod** command shall create a special file named *name* of the given *type*.

The *type* shall be one of the following:

b

creates a block (buffered) special file with the specified *major* and *minor* device numbers.

c, u

creates a character (unbuffered) special file with the specified *major* and *minor* device numbers.

p

creates a FIFO.

Options

-m *mode*, --mode=*mode*

create the special file with file access permissions set as described in *mode*. The permissions may be any absolute value (i.e. one not containing '+' or '-') acceptable to the **chmod** command.

--version

output version information and exit.

This option may be deprecated in a future release of this specification.

If *type* is pparameter, *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 insufficently 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 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

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. A unique name is generated, but the temporary file shall be unlinked before **mktemp** exits. Use of this option is not encouraged.

more

Name

`more` — display files on a page-by-page basis

Description

more is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

The **more** command need not respect the `LINES` and `COLUMNS` environment variables.

The following additional options may be supported:

-*num*

specifies an integer which is the screen size (in lines).

+*num*

starts at line number *num*.

+/*pattern*

Start at the first line matching the pattern, equivalent to executing the search forward (`/`) command with the given pattern immediately after opening each file.

The following options from ISO POSIX (2003) may behave differently:

-e

has unspecified behavior.

-i

has unspecified behavior.

-n

has unspecified behavior.

-p

Either clear the whole screen before displaying any text (instead of the usual scrolling behavior), or provide the behavior specified by ISO POSIX (2003). In the latter case, the syntax is "`-p` *command*".

-t

has unspecified behavior.

The **more** command need not support 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
ZZ

Rationale

The +*num* and +/*string* options are deprecated in SUSv2, and have been removed in ISO POSIX (2003); however this specification continues to specify them because the publicly available `util-linux` package does not support the replacement (-p *command*). The +*command* option as found in SUSv2 is more general than is specified here, but the `util-linux` package appears to only support the more specific +*num* and +/*string* forms.

mount

Name

`mount` — mount a file system

Synopsis

mount [-hV]
mount [-a] [-fFnrsvw] [-t *`vfstype`*]
mount [-fnrsvw] [-o *`options`* [,...]] [device | dir]
mount [-fnrsvw] [-t *`vfstype`*] [-o options] device dir

Description

As described in ISO POSIX (2003), all files in the system are organized in a directed graph, known as the file hierarchy, rooted at `/`. These files can be spread out over several underlying devices. The **mount** command shall attach the file system found on some underlying device to the file hierarchy.

Options

-v

invoke verbose mode. The **mount** command shall provide diagnostic messages on `stdout`.

-a

mount all filesystems (of the given types) mentioned in `/etc/fstab`.

-F

If the *`-a`* option is also present, fork a new incarnation of **mount** for each device to be mounted. This will do the mounts on different devices or different NFS servers in parallel.

-f

cause everything to be done except for the actual system call; if it's not obvious, this `fakes' mounting the file system.

-n

mount without writing in `/etc/mtab`. This is necessary for example when `/etc` is on a read-only file system.

-s

ignore **mount** options not supported by a filesystem type. Not all filesystems support this option.

-r

mount the file system read-only. A synonym is *`-o ro`*.

-w

mount the file system read/write. (default) A synonym is *`-o rw`*.

-L label

If the file `/proc/partitions` is supported, mount the partition that has the specified label.

-U uuid

If the file `/proc/partitions` is supported, mount the partition that has the specified uuid.

-t vfstype

indicate a file system type of *`vfstype`*.

More than one type may be specified in a comma separated list. The list of file system types can be prefixed with `no` to specify the file system types on which no action should be taken.

-o

options are specified with a *`-o`* flag followed by a comma-separated string of options. Some of these options are only useful when they appear in the `/etc/fstab` file. The following options apply to any file system that is being mounted:

async

perform all I/O to the file system asynchronously.

atime

update inode access time for each access. (default)

auto

in `/etc/fstab`, indicate the device is mountable with *`-a`*.

defaults

use default options: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`.

dev

interpret character or block special devices on the file system.

exec

permit execution of binaries.

noatime

do not update file access times on this file system.

noauto

in `/etc/fstab`, indicates the device is only explicitly mountable.

nodev

do not interpret character or block special devices on the file system.

noexec

do not allow execution of any binaries on the mounted file system.

nosuid

do not allow set-user-identifier or set-group-identifier bits to take effect.

nouser

forbid an unprivileged user to mount the file system. (default)

remount

remount an already-mounted file system. This is commonly used to change the mount options for a file system, especially to make a read-only file system writable.

ro

mount the file system read-only.

rw

mount the file system read-write.

suid

allow set-user-identifier or set-group-identifier bits to take effect.

sync

do all I/O to the file system synchronously.

user

allow an unprivilieged user to mount the file system. This option implies the options `noexec`, `nosuid`, `nodev` unless overridden by subsequent options.

LSB Deprecated Options

The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should only use the non-LSB-deprecated behaviors.

-V

output version and exit.

msgfmt

Name

`msgfmt` — create a message object from a message file

Synopsis

msgfmt [options...] *`filename`*...

Description

The **msgfmt** command generates a binary message catalog from a textual translation description. Message catalogs, or message object files, are stored in files with a `.mo` extension.

> The format of message object files is not guaranteed to be portable. Message catalogs should always be generated on the target architecture using the **msgfmt** command.

The source message files, otherwise known as portable object files, have a `.po` extension.

The *`filename`* operands shall be portable object files. The `.po` file contains messages to be displayed to users by system utilities or by application programs. The portable object files are text files, and the messages in them can be rewritten in any language supported by the system.

If any *`filename`* is `-`, a portable object file shall be read from the standard input.

The **msgfmt** command interprets data as characters according to the current setting of the `LC_CTYPE` locale category.

Options

-c
--check

Detect and diagnose input file anomalies which might represent translation errors. The `msgid` and `msgstr` strings are studied and compared. It is considered abnormal that one string starts or ends with a newline while the other does not.

If the message is flagged as `c-format` (see Comment Handling), check that the `msgid` string and the `msgstr` translation have the same number of `%` format specifiers, with matching types.

-D *`directory`*
--directory=*`directory`*

Add directory to list for input files search. If *`filename`* is not an absolute pathname and *`filename`* cannot be opened, search for it in *`directory`*. This option may be repeated. Directories shall be searched in order, with the leftmost *`directory`* searched first.

-f
--use-fuzzy

Use entries marked as `fuzzy` in output. If this option is not specified, such entries are not included into the output. See Comment Handling below.

-o *output-file*
--output-file=*output-file*

Specify the output file name as `output-file`. If multiple domains or duplicate msgids in the `.po` file are present, the behavior is unspecified. If output-file is `-`, output is written to standard output.

-S
--strict

Ensure that all output files have a `.mo` extension. Output files are named either by the *`-o`* (or *`--output-file`*) option, or by domains found in the input files.

-v
--verbose

Print additional information to the standard error, including the number of translated strings processed.

Operands

The *`filename`* operands are treated as portable object files. The format of portable object files is defined in EXTENDED DESCRIPTION.

Standard Input

The standard input is not used unless a *`filename`* operand is specified as "-".

Environment Variables

LANGUAGE

Specifies one or more locale names.

LANG

Specifies locale name.

LC_ALL

Specifies locale name for all categories. If defined, overrides LANG, LC_CTYPE and LC_MESSAGES.

LC_CTYPE

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Specifies messaging locale, and if present overrides LANG for messages.

Standard Output

The standard output is not used unless the option-argument of the *`-o`* option is specified as `-`.

Extended Description

The format of portable object files (`.po` files) is defined as follows. Each `.po` file contains one or more lines, with each line containing either a comment or a statement. Comments start the line with a hash mark (#) and end with the newline character. Empty lines, or lines containing only white-space, shall be ignored. Comments can in certain circumstances alter the behavior of **msgfmt**. See Comment Handling below for details on comment processing. The format of a statement is:

```
directive value
```

Each `directive` starts at the beginning of the line and is separated from `value` by white space (such as one or more space or tab characters). The `value` consists of one or more quoted strings separated by white space. If two or more strings are specified as `value`, they are normalized into single string using the string normalization syntax specified in ISO C (1999). The following directives are supported:

```
domain domainname
```

```
msgid message_identifier
```

```
msgid_plural untranslated_string_plural
```

```
msgstr message_string
```

```
msgstr[n] message_string
```

The behavior of the domain directive is affected by the options used. See OPTIONS for the behavior when the *`-o`* option is specified. If the *`-o`* option is not specified, the behavior of the `domain` directive is as follows:

1. All msgids from the beginning of each `.po` file to the first `domain` directive are put into a default message object file, messages (or `messages.mo` if the *`--strict`* option is specified).
2. When **msgfmt** encounters a `domain domainname` directive in the `.po` file, all following *`msgids`* until the next `domain` directive are put into the message object file `domainname` (or `domainname.mo` if *`--strict`* option is specified).
3. Duplicate *`msgids`* are defined in the scope of each domain. That is, a *`msgid`* is considered a duplicate only if the identical *`msgid`* exists in the same domain.
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, ...) 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:

Table 3-1. Escape Sequences

`\n`	newline
`\t`	tab
`\v`	vertical tab

\b	backspace
\r	carriage return
\f	formfeed
\\	backslash
\"	double quote
\ddd	octal bit pattern
\xHH	hexadecimal bit pattern

Comment Handling

Comments are introduced by a #, and continue to the end of the line. The second character (i.e. the character following the #) has special meaning. Regular comments should follow a space character. Other comment types include:

```
# normal-comments
```

```
#. automatic-comments
```

```
#: reference...
```

```
#, flag
```

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 shows that the following `msgstr` string might not be a correct translation. 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.
>
> 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 `c-format` flag indicates that the `msgid` string is used as format string by `printf`-like functions. If the `c-format` flag is given for a string the **msgfmt** utility 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` 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

The following exit values are returned:

0

: Successful completion.

>0

: An error occurred.

Application Usage

Neither **msgfmt** nor any `gettext` function imposes a limit on the total length of a message. Installing message catalogs under the C locale is pointless, since they are ignored for the sake of efficiency.

Examples

Example 1: Examples of creating message objects from message files.

In this example `module1.po`, `module2.po` and `module3.po` are portable message object files.

```
example% cat module1.po

# default domain "messages"

msgid "message one"

```

```
804   msgstr "mensaje número uno"
805
806   #
807
808   domain "help_domain"
809
810   msgid "help two"
811
812   msgstr "ayuda número dos"
813
814   #
815
816   domain "error_domain"
817
818   msgid "error three"
819
820   msgstr "error número tres"
821
822   example% cat module2.po
823
824   # default domain "messages"
825
826   msgid "message four"
827
828   msgstr "mensaje número cuatro"
829
830   #
831
832   domain "error_domain"
833
834   msgid "error five"
835
836   msgstr "error número cinco"
837
838   #
839
840   domain "window_domain"
841
842   msgid "window six"
843
844   msgstr "ventana número seises"
845   example% cat module3.po
846
847   # default domain "messages"
848
849   msgid "message seven"
850
851   msgstr "mensaje número siete"
```

The following command will produce the output files `messages`, `help_domain`, and `error_domain`.

```
example% msgfmt module1.po
```

The following command will produce the output files `messages`, `help_domain`, `error_domain`, and `window_domain`.

```
example% msgfmt module1.po module2.po
```

The following example will produce the output file `hello.mo`.

```
example% msgfmt -o hello.mo module3.po
```

newgrp

Name

`newgrp` — change group ID

Synopsis

newgrp [group]

Description

The **newgrp** command is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

The `-l` option specified in ISO POSIX (2003) need not be supported.

od

Name

od — dump files in octal and other formats

Synopsis

od [-abcdfilox] [-w *width* | --width-*width*] [-v] [-A *address_base*] [-j *skip*] [-n *count*] [-t *type_string*] [file...]
od --traditional [options] [file] [[+]offset [.] [b]] [[+]label [.] [b]]

Description

od is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

-w*width*, --width[=*width*]

each output line is limited to *width* bytes from the input.

--traditional

accepts arguments in traditional form.

> The XSI optional behavior described in ISO POSIX (2003) is not supported unless the *--traditional* option is also specified.

Pre-POSIX and XSI Specifications

The LSB supports option intermixtures with the following pre-POSIX and XSI options:

-a

is equivalent to *-t a*, selects named characters.

-b

is equivalent to *-t o1*, selects octal bytes.

-c

is equivalent to *-t c*, selects characters.

-d

is equivalent to *-t u2*, selects unsigned decimal two byte units.

-f

is equivalent to *-t fF*, selects floats.

-i

888 is equivalent to `-t d2`, selects decimal two byte units.

889 This usage may change in future releases; portable applications should use `-t d2`.

890 -l

891 is equivalent to `-t d4`, selects decimal longs.

892 -o

893 is equivalent to `-t o2`, selects octal two byte units.

894 -x

895 is equivalent to `-t x2`, selects hexadecimal two byte units.

896 Note that the XSI option `-s` need not be supported.

Traditional Usage

897 If the `--traditional` is specified, there may be between zero and three operands specified.

898 If no operands are specified, then od shall read the standard input.

899 If there is exactly one operand, and it is an offset of the form `[+]offset[.][b]`, then it shall be interpreted as
900 specified in ISO POSIX (2003). The file to be dumped shall be the standard input.

901 If there are exactly two operands, and they are both of the form `[+]offset[.][b]`, then the first shall be an treated as
902 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
903 output line produced for each input block shall be preceded by the input offset, cumulative across input files, of the
904 next byte to be written, followed by the label, in parentheses. The label shall increment in the same manner as the
905 offset.

906 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

`passwd` — change user password

Synopsis

passwd [-x max] [-n min] [-w warn] [-i inact] name
passwd {-l | -u} name

Description

passwd changes passwords for user and group accounts. A normal user may only change the password for their own account, the super user may change the password for any account. **passwd** also changes password expiry dates and intervals. Applications may not assume the format of prompts and anticipated input for user interaction, because they are unspecified.

Options

-x max

: sets the maximum number of days a password remains valid.

-n min

: sets the minimum number of days before a password may be changed.

-w warn

: sets the number of days warning the user will receive before their password will expire.

-i inactive

: disables an account after the password has been expired for the given number of days.

-l

: disables an account by changing the password to a value which matches no possible encrypted value.

-u

: re-enables an account by changing the password back to its previous value.

patch

Name

`patch` — apply a diff file to an original

Description

patch is as specified in ISO POSIX (2003), but with extensions as listed below.

Extensions

--binary

reads and write all files in binary mode, except for standard output and /dev/tty. This option has no effect on POSIX-compliant systems.

-u, --unified

interprets the patch file as a unified context diff.

pidof

Name

`pidof` — find the process ID of a running program

Synopsis

pidof [-s] [-x] [-o omitpid...] program...

Description

Return the process ID of a process which is running the program named on the command line.

Options

-s

instructs the program to only return one pid.

-x

causes the program to also return process id's of shells running the named scripts.

-o

omits processes with specified process id.

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 ISO POSIX (2003), but with differences as listed below.

Differences

-n increment

has unspecified behavior.

sed

Name

`sed` — stream editor

Description

sed is as specified in ISO POSIX (2003), but with differences as listed below.

LSB Differences

Certain aspects of internationalized regular expressions are optional; see Internationalization and Regular Expressions>.

sendmail

Name

`sendmail` — an electronic mail transport agent

Synopsis

sendmail [options] [address...]

Description

To deliver electronic mail (email), applications shall support the interface provided by /usr/sbin/sendmail (described here). This interface shall be the default delivery method for applications.

This program sends an email message to one or more recipients, routing the message as necessary. This program is not intended as a user interface routine.

With no options, **sendmail** reads its standard input up to an end-of-file or a line consisting only of a single dot and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the syntax and contents of the addresses.

It is recommended that applications use as few options as necessary, none if possible.

Some agents allow aliasing on the local system to be prevented by preceding the address with a backslash.

The format of messages shall be as defined in RFC 2822.

Options

-bm

: reads mail from standard input and delivers to the recipient addresses. This is the default mode of operation.

-bp

: lists information about messages currently in the input mail queue.

-bs

: uses the SMTP protocol as described in RFC 2821; reads SMTP commands on standard input and writes SMTP responses on standard output.

  Note that RFC 2821 specifies \r\n (CR-LF) be used at the end of each line, but pipes almost always use \n (LF) 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 before the \n is silently discarded.

-F fullname

: explicitly sets the full name of the sender for incoming mail unless the message already contains a From: message header.

  If the user running **sendmail** is not sufficiently trusted, then the actual sender may be indicated in the message, depending on the behavior of the agent.

-f name

explicitly sets the envelope sender address for incoming mail. If there is no From: header, the address specified in the From: header will also be set.

If the user running **sendmail** is not sufficiently trusted, then the actual sender will be indicated in the message.

-i

ignores dots alone on lines by themselves in incoming messages. If -bs is also used, the behavior is unspecified.

-odb

delivers any mail in background, if supported; otherwise ignored.

-odf

delivers any mail in foreground, if supported; otherwise ignored.

-oem or -em

mails errors back to the sender. (default)

-oep or -ep

writes errors to the standard error output.

-oeq or -eq

does not send notification of errors to the sender. This only works for mail delivered locally.

-oi

is equivalent to -i.

-om

indicates that the sender of a message should receive a copy of the message if the sender appears in an alias expansion. Ignored if aliases are not supported.

-t

reads the message to obtain recipients from the To:, Cc:, and Bcc: headers in the message instead of from the command arguments. If a Bcc: header is present, it is removed from the message unless there is no To: or Cc: header, in which case a Bcc: header with no data is created, in accordance with RFC 2822.

If there are any arguments, they specify addresses to which the message is not to be delivered. That is, the argument addresses are removed from the recipients list obtained from the headers. Note: some agents implement this behavior in reverse, adding addresses instead of removing them. Others may disallow addresses in argument list. Therefore, applications should not put addresses in the argument list if -t is used.

This option is sometimes ignored when not in -bm mode (the default).

Exit status

0

successful completion on all addresses. This does not indicate successful delivery.

>0

there was an error.

Notes/Rationale

This page is believed to reflect functionality provided by smail, exim and other implementations, not just the **sendmail** implementation.

shutdown

Name

`shutdown` — bring the system down

Synopsis

/sbin/shutdown [-t sec] [-arkhcfF] time [warning-message]

Description

shutdown brings the system down in a secure way. All logged-in users are notified that the system is going down, and login(1) is blocked. It is possible to shut the system down immediately or after a specified delay. All processes are first notified that the system is going down by the signal SIGTERM. If neither the -h or the -r argument is used, then the default behavior is to take the system to runlevel one where administrative tasks can be run.

Standard Options

-a

uses /etc/shutdown.allow.

-t sec

tells init(8) to wait sec seconds between sending processes the warning and the kill signal, before changing to another runlevel.

-k

doesn't really shutdown; only sends the warning messages to everybody.

-r

reboots after shutdown.

-h

halts after shutdown. Powering off after halting is unspecified.

-f

skips fsck on reboot.

-F

forces fsck on reboot.

-c

cancels an already running **shutdown**. With this option, it is of course not possible to give the time argument, but you can enter a explanatory message on the command line that will be sent to all users.

time

specifies when to shut down.

The time argument can have different formats. First, it can be an absolute time in the format hh:mm, in which hh 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 which m is the number of minutes to wait. The word now is an alias for +0.

If **shutdown** is called with a delay, it creates the advisory file /etc/nologin which causes programs such as login(1) to not allow new user logins. **shutdown** only removes this file if it is stopped before it can signal init (i.e. it is cancelled or something goes wrong). Otherwise it is the responsibility of the system shutdown or startup scripts to remove this file so that users can login.

warning-message

specifies message to send all users.

su

Name

su — change user ID or become super-user

Synopsis

su [options] [-] [username [ARGS]]

Description

su is used to become another user during a login session. Invoked without a username, **su** defaults to becoming the super user. The optional argument - may be used to provide an environment similar to what the user would expect had the user logged in directly.

The user will be prompted for a password, if appropriate. Invalid passwords will produce an error message. All attempts, both valid and invalid, are logged to detect abuses of the system. Applications may not assume the format of prompts and anticipated input for user interaction, because they are unspecified.

An optional command can be executed. This is done by the shell specified in /etc/passwd for the target user unless the -s or -m options are used. Any arguments supplied after the username will be passed to the invoked shell (shell shall support the -c command line option in order for a command to be passed to it).

The current environment is passed to the new shell. The value of $PATH is reset to /bin:/usr/bin for normal users, or /sbin:/bin:/usr/sbin:/usr/bin for the super user. This may be changed with the ENV_PATH and ENV_SUPATH definitions in /etc/login.defs. When using the -m or -p options, the user's environment is not changed.

A subsystem login is indicated by the presense of a "*" as the first character of the login shell. The given home directory will be used as the root of a new filesystem which the user is actually logged into.

Standard Options

-

: makes this a login shell.

-c, --comand=command

: passes command to the invoked shell. It is passed directly to the invoked shell (using the shell's -c option), so its syntax is whatever that shell can accept.

-m, -p, --preserve-environment

: does not reset environment variables, and keeps the same shell if it is present in /etc/shells.

-s, --shell=shell

: uses shell instead of the default in /etc/passwd. The shell specified shall be present in /etc/shells.

sync

Name

`sync` — flush filesystem buffers

Synopsis

sync

Description

Force changed blocks to disk, update the super block.

tar

Name

`tar` — file archiver

Description

tar is as specified in SUSv2, but with differences as listed below.

Differences

Certain aspects of internationalized filename globbing are optional; see Internationalization and Pattern Matching Notation>.

-h

doesn't dump symlinks; dumps the files they point to.

-z

filters the archive through **gzip**.

umount

Name

`umount` — unmount file systems

Synopsis

umount [-hV]
umount -a [-nrv] [-t vfstype]
umount [-nrv] device | dir

Description

umount detaches the file system(s) mentioned from the file hierarchy. A file system is specified by giving the directory where it has been mounted.

Standard Options

-v

invokes verbose mode.

-n

unmounts without writing in /etc/mtab.

-r

tries to remount read-only if unmounting fails.

-a

unmounts all of the file systems described in /etc/mtab except for the proc filesystem.

-t vfstype

indicates that the actions should only be taken on file systems of the specified type. More than one type may be specified in a comma separated list. The list of file system types can be prefixed with no to specify the file system types on which no action should be taken.

-f

forces unmount (in case of an unreachable NFS system).

LSB Deprecated Options

The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should only use the non-LSB-deprecated behaviors.

-V

print version and exits.

useradd

Name

`useradd` — create a new user or update default new user information

Synopsis

```
useradd [-c comment] [-d home_dir]
        [-g initial_group] [-G group[,...]]
        [-m [-k skeleton_dir]] [-p passwd] [-r]
        [-s shell] [-u uid [ -o]] login

useradd -D [-g default_group] [-b default_home]
```

```
[-s default_shell]
```

Description

When invoked without the -D option, **useradd** creates a new user account using the values specified on the command line and the default values from the system. The new user account will be entered into the system files as needed, the home directory will be created, and initial files copied, depending on the command line options.

When invoked with the -D option, **useradd** will either display the current default values, or update the default values from the command line. If no options are specified, **useradd** displays the current default values.

Standard Options

-c comment

specifies the new user's password file comment field value.

-d home_dir

creates the new user using home_dir as the value for the user's login directory. The default is to append the login name to default_home and use that as the login directory name.

-g initial_group

specifies the group name or number of the user's initial login group. The group name shall exist. A group number shall refer to an already existing group. If -g is not specified, the implementation will follow the normal user default for that system. This may create a new group or choose a default group that normal users are placed in. Applications which require control of the groups into which a user is placed should specify -g.

-G group,[...]

specifies a list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the -g option. The default is for the user to belong only to the initial group.

-m [-k skeleton_dir]

specifies the user's home directory will be created if it does not exist. The files contained in skeleton_dir will be copied to the home directory if the -k option is used, otherwise the files contained in /etc/skel will be used instead. Any directories contained in skeleton_dir or /etc/skel will be created in the user's home directory as well. The -k option is only valid in conjunction with the -m option. The default is to not create the directory and to not copy any files.

-p passwd

is the encrypted password, as returned by crypt(3). The default is to disable the account.

-r

creates a system account, that is, a user with a UID in the range reserved for system account users. If there is not a UID free in the reserved range the command will fail.

-s shell

specifies the name of the user's login shell. The default is to leave this field blank, which causes the system to select the default login shell.

-u uid [-o]

specifies the numerical value of the user's ID. This value shall be unique, unless the -o option is used. The value shall be non-negative. The default is the smallest ID value greater than 499 which is not yet used.

Change Default Options

-b default_home

specifies the initial path prefix for a new user's home directory. The user's name will be affixed to the end of default_home to create the new directory name if the -d option is not used when creating a new account.

-g default_group

specifies the group name or ID for a new user's initial group. The named group shall exist, and a numerical group ID shall have an existing entry.

-s default_shell

specifies the name of the new user's login shell. The named program will be used for all future new user accounts.

-c comment

specifies the new user's password file comment field value.

Application Usage

The -D option will typically be used by system administration packages. Most applications should not change defaults which will affect other applications and users.

userdel

Name

`userdel` — delete a user account and related files

Synopsis

userdel [-r] login

Description

Delete the user account named *`login`*. If there is also a group named *`login`*, this command may delete the group as well, or may leave it alone.

Options

-r

removes files in the user's home directory along with the home directory itself. Files located in other file system will have to be searched for and deleted manually.

usermod

Name

`usermod` — modify a user account

Synopsis

```
usermod [-c comment] [-d home_dir [ -m]]
        [-g initial_group] [-G group[,...]]
        [-l login_name] [-p passwd]
```

```
        [-s shell] [-u uid [ -o]] login
```

Options

-c comment

specifies the new value of the user's password file comment field.

-d home_dir

specifies the user's new login directory. If the -m option is given the contents of the current home directory will be moved to the new home directory, which is created if it does not already exist.

-g initial_group

specifies the group name or number of the user's new initial login group. The group name shall exist. A group number shall refer to an already existing group.

-G group,[...]

specifies a list of supplementary groups which the user is also a member of. Each group is separated from the next by a comma, with no intervening whitespace. The groups are subject to the same restrictions as the group given with the -g option. If the user is currently a member of a group which is not listed, the user will be removed from the group.

-l login_name

changes the name of the user from login to login_name. Nothing else is changed. In particular, the user's home directory name should probably be changed to reflect the new login name.

-p passwd

is the encrypted password, as returned by crypt(3).

-s shell

specifies the name of the user's new login shell. Setting this field to blank causes the system to select the default login shell.

-u uid [-o]

specifies the numerical value of the user's ID. This value shall be unique, unless the -o option is used. The value shall be non-negative. Any files which the user owns and which are located in the directory tree rooted at the user's home directory will have the file user ID changed automatically. Files outside of the user's home directory shall be altered manually.

xargs

Name

`xargs` — build and execute command lines from standard input

Description

xargs is as specified in ISO POSIX (2003), but with differences as listed below.

Differences

-E

has unspecified behavior.

-I

has unspecified behavior.

-L

has unspecified behavior.

IV. Execution Environment

Chapter 4. File System Hierarchy

An LSB conforming implementation shall provide the mandatory portions of the filesystem hierarchy specified in the Filesystem Hierarchy Standard (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) required by the LSB 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_ISVTXT` bit (otherwise known as the "sticky bit").

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 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

In order to implement common security policies it is strongly advisable for applications to use the 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

In general, applications should not depend on running as a privileged user. This specification uses the term "appropriate privilege" throughout to identify operations that cannot be achieved without some special granting of additional 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 with appropriate privilege, as this makes security auditing harder or even impossible.

5.2.6. Changing permissions

The application shall not change permissions of files and directories that do not belong to its own package. Should an application require that certain files and directories not directly belonging to the package have a particular ownership, the application shall document this requirement, and may fail during installation if the permissions on these files is inappropriate.

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 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

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.

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 POSIX (2003)*:

_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 POSIX (2003)*.

The `access()` function 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 POSIX (2003)*.

Calling `unlink()` on a directory shall fail. Calling `link()` specifying a directory as the first argument shall fail. See also unlink.

> 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 POSIX (2003).

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()`.

Chapter 7. Localization

In order to install a message catalog, the installation procedure shall supply the message catalog in a format readable by the **msgfmt** utility, which shall be invoked to compile the message catalog 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 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 ISO POSIX (2003).

7.1. Regular Expressions

Utilities that process regular expressions shall support Basic Regular Expressions and Extended Regular Expressions as specified in 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. Pattern Matching Notation

Utilities that perform filename pattern matching (also known as Filename Globbing) shall do it as specified in ISO POSIX (2003), Pattern Matching Notation, with the following exceptions:

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

In addition to the individual user `crontab` files specified by ISO POSIX (2003) stored under `/var/spool/cron`, 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 wishes to install a job that has to be executed periodically, it shall place a file in one of the following directories:

```
/etc/cron.daily
/etc/cron.weekly
/etc/cron.monthly
```

As these directory names 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 files in these directories.

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.

The scripts in these directories should check if all necessary programs are installed before they try to execute them. Otherwise, problems will arise if a package 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 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 `'.'`. 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.

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
200-254	reserved

In the case of init script commands other than "status" (i.e., "start", "stop", "restart", "try-restart", "reload", and "force-reload"), the init script shall return an exit status of zero if the action described by the argument has been successful. Otherwise, the exit status shall be non-zero, as defined below. In addition to straightforward success, the following situations are also to be considered successful:

- restarting a service (instead of reloading it) with the "force-reload" argument
- running "start" on a service already running
- running "stop" on a service already stopped or not running
- running "restart" on a service already stopped or not running
- running "try-restart" on a service already stopped or not running

In case of an error, while processing any init script action except for "status", the init script shall print an error message 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
200-254	reserved

Error and status messages should be printed with the logging functions such as log_failure_msg and so on. Scripts may write to standard error or standard output, but implementations need not present text written to standard error/output to the user or do anything else with it.

Since init files may be run manually by a system administrator with non-standard environment variable values for PATH, USER, LOGNAME, etc. init files shall not depend on the values of these environment variables. They should set them to some known/default values if they are needed.

8.3. Comment Conventions for Init Scripts

LSB applications which need to execute script(s) at bootup and/or shutdown may provide one or more init.d files. These files are installed by the install_initd program described below, which copies it into a standard directory and makes whatever other adjustments (creation of symlinks, creation of entries in a database, etc.) are necessary so that the script can be run at boot-time. [1]

In the init.d file, information about the shell script shall be delimited by the lines "### BEGIN INIT INFO" and "### END INIT INFO". These delimiter lines may containg trailing whitespace, which shall be ignored. Inside this block 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-xyzzydecl".

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 eligble 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 mulitple 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:

```
/usr/lib/lsb/install_initd /etc/init.d/example.com-coffeed
```

The install_initd program shall return an exit status of zero if the init.d file has been successfully installed or if the the init.d file was already installed. If the required boot facilities cannot be fulfilled an exit status of one shall be returned and the init.d file shall not be installed.

When a software package is removed, the package's preuninstall script shall call /usr/lib/lsb/remove_initd and pass the pathname to the /etc/init.d file. The package manager is still responsible for removing the /etc/init.d file; the remove_initd program is provided in case the distribution needs to clean up any other modifications in the distribution's boot script system that might have been made by the install_initd program. For example:

```
/usr/lib/lsb/remove_initd /etc/init.d/example.com-coffeed
```

The remove_initd program shall return an exit status of zero if the init.d file has been successfully removed or if the the 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, an exit status of one shall be returned and the init.d file shall remained installed.

There should be a tool available to the user (e.g., RedHat's chkconfig) which can be used by the system administrator to easily manipulate at which init levels a particular init.d script is started or stopped. This specification currently does not specify such an interface, however.

8.5. Run Levels

The following run levels are specified for use by the "Default-Start:" and "Default-Stop:" specifiers as defined by the section *Comment Conventions for Init Scripts*>. Many LSB run-time environments commonly use these run level definitions, and in the absence of other considerations, providers of run-time environments are strongly encouraged to follow this convention to provide consistency for system administrators who need to work with multiple distributions. However, it is not required that LSB-compliant run-time environments use these run levels; the distribution-provided install_initd script may map the run levels specified below to whatever distribution-specified run levels are most 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

Boot facilities are used to indicate dependencies in init scripts, as defined in a previous section. Facility names that begin with a dollar sign ('$') are system facility names, defined by the LSB, and SHALL be provided by distributions. [5] LSB applications shall not provide facilities that begin with a dollar sign. This document defines the following 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 running[6]
$portmap	daemons providing SunRPC/ONCRPC portmapping service[7] (if present) are running
$remote_fs	all remote filesystems are mounted[8].
$syslog	system logger is operational
$time	the system time has been set [9]

Other (non-system) facilities may be defined by other LSB applications. These facilities shall be named using the same conventions defined for naming init.d script names. Commonly, the facility provided by an LSB application init.d script will have the same name as the name assigned to the init.d script.

8.7. Script Names

Since the init.d scripts shall live in a single directory, they shall come from a single namespace. Three means of assigning names from this namespace are available:

- Assigned namespace. This namespace consists of names which only use the character set [a-z0-9]. This space is desirable for scripts which system administrators may often wish to run manually: e.g., "/etc/init.d/named restart" In order to avoid conflicts these init.d 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 to 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: [hier1]-[hier2]-...-[name], where name is again taken 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 '.' (e.g., "debian.org", "staroffice.sun.com"). 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.

In general, if a package or some system function is likely to be used on multiple systems, the package developers or the distribution SHOULD get a registered name through LANANA, and distributions should strive to use the same name whenever possible. For applications which may not be "core" or may not be commonly installed, the hierarchical namespace may be more appropriate. An advantage to the hierarchical namespace is that there is no need to consult with the LANANA before obtaining an assigned name.

Short names are highly desirable, since many system administrators like to use them to manually start and stop services. Given this, they should be standardized on a per-package basis. This is the rationale behind having a LANANA organization to assign these names. The LANANA may be called upon to handle other namespace issues, such as package/prerequisites naming (which is essential to making prerequisites to work correctly).

8.8. Init Script Functions

Each LSB-compliant init.d script shall source the file `/lib/lsb/init-functions`. This file shall cause the following shell script commands to be defined. This can be done either by adding a directory to the PATH variable which defines these commands, or by defining sh aliases. While the distribution-provided aliases may choose to use shell extensions (at the distribution's option), the LSB init.d files themselves should only depend in shell features as defined by the LSB.

The **start_daemon**, **killproc** and **pidofproc** functions shall use this algorithm for determining the status and the pid(s) of the specified program. They shall read the pidfile specified or otherwise `/var/run/`*`basename`*`.pid` and use the pid(s) herein when determining whether a program is running. The method used to determine the status is implementation defined, but should allow for non-binary programs. [10] Compliant implementations may use other mechanisms besides those based on pidfiles, unless the -p pidfile option has been used. Compliant applications should not rely on such mechanisms and should always use a pidfile. When a program is stopped, it should delete its pidfile. 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 This runs the specified program as a daemon.

[args]	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 *-f* option is given. The *-n* option specifies a nice level. See nice(1). 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.
killproc [-p pidfile] pathname [signal]	This stops the specified program. The program is found using the algorithm given above. If a signal is specified, using the *-signal_name* or *-signal_number* 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.
pidofproc [-p pidfile] pathname	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.
log_success_msg "message"	This requests the distribution to print a success message. The message should be relatively short; no more than 60 characters is highly desirable.
log_failure_msg "message"	This requests the distribution to print a failure message. The message should be relatively short; no more than 60 characters is highly desirable.
log_warning_msg "message"	This requests the distribution to print a warning message. The message should be relatively short; no more than 60 characters is highly desirable.

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

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 all appropriate privileges
bin	bin	Legacy UID/GID[a]
daemon	daemon	Legacy UID/GID[b]

Notes:

a. The 'bin' UID/GID is included for compatibility with legacy applications. New applications should no longer use the 'bin' UID/GID.

b. 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 ugidd(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 dynamic 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. libc

1 The behaviour of the interfaces in this library is specified by the following Standards.

Large File Support
this specification
SUSv2
ISO POSIX (2003)
SVID Issue 3
2 SVID Issue 4

3 **Table A-1. libc Function Interfaces**

_Exit(GLIBC_2.1.1)[1]	getrlimit(GLIBC_2.1.1)[1]	sigandset(GLIBC_2.1.1)[1]
_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]

__isinff[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]
__mempcpy(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]
__wcstol_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_sperror(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]	towlower(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]	ttyname(GLIBC_2.0)[1]
endgrent(GLIBC_2.0)[1]	munlock(GLIBC_2.0)[1]	ttyname_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]	nrand48(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]	vsnprintf(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]	wcscasecmp(GLIBC_2.2.3)[1]
fopen(GLIBC_2.1)[1]	rand(GLIBC_2.1)[1]	wcscat(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]	wcscmp(GLIBC_2.0)[1]
fpathconf(GLIBC_2.0)[1]	random_r(GLIBC_2.0)[1]	wcscoll(GLIBC_2.0)[1]
fprintf(GLIBC_2.0)[1]	read(GLIBC_2.0)[1]	wcscpy(GLIBC_2.0)[1]
fputc(GLIBC_2.0)[1]	readdir(GLIBC_2.0)[1]	wcscspn(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]	regexec(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]
gcvt(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]	writev(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-2. libc Data Interfaces

__daylightID_STD_46_LSB	__timezoneID_STD_46_LSB	_sys_errlistID_STD_46_LSB
__environID_STD_46_LSB	__tznameID_STD_46_LSB	

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

crypt(GLIBC_2.0)[1]	encrypt(GLIBC_2.0)[1]	setkey(GLIBC_2.0)[1]

A.3. libdl

The behaviour of the interfaces in this library is specified by the following Standards.

this specification
ISO POSIX (2003)

Table A-4. libdl Function Interfaces

dladdr(GLIBC_2.0)[1]	dlerror(GLIBC_2.0)[1]	dlsym(GLIBC_2.0)[1]
dlclose(GLIBC_2.0)[1]	dlopen(GLIBC_2.0)[1]	

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

acos(GLIBC_2.0)[1]	csinhl(GLIBC_2.0)[1]	log(GLIBC_2.0)[1]
acosf(GLIBC_2.0)[1]	csinl(GLIBC_2.0)[1]	log10(GLIBC_2.0)[1]
acosh(GLIBC_2.0)[1]	csqrt(GLIBC_2.0)[1]	log10f[1]
acoshf(GLIBC_2.0)[1]	csqrtf(GLIBC_2.0)[1]	log10l[1]
acoshl(GLIBC_2.0)[1]	csqrtl(GLIBC_2.0)[1]	log1p(GLIBC_2.0)[1]
acosl(GLIBC_2.0)[1]	ctan(GLIBC_2.0)[1]	logb(GLIBC_2.0)[1]

asin(GLIBC_2.0)[1]	ctanf(GLIBC_2.0)[1]	logf[1]
asinf(GLIBC_2.0)[1]	ctanh(GLIBC_2.0)[1]	logl[1]
asinh(GLIBC_2.0)[1]	ctanhf(GLIBC_2.0)[1]	lrint(GLIBC_2.0)[1]
asinhf(GLIBC_2.0)[1]	ctanhl(GLIBC_2.0)[1]	lrintf(GLIBC_2.0)[1]
asinhl(GLIBC_2.0)[1]	ctanl(GLIBC_2.0)[1]	lrintl(GLIBC_2.0)[1]
asinl(GLIBC_2.0)[1]	dremf(GLIBC_2.0)[1]	lround(GLIBC_2.0)[1]
atan(GLIBC_2.0)[1]	dreml(GLIBC_2.0)[1]	lroundf(GLIBC_2.0)[1]
atan2(GLIBC_2.0)[1]	erf(GLIBC_2.0)[1]	lroundl(GLIBC_2.0)[1]
atan2f(GLIBC_2.0)[1]	erfc(GLIBC_2.0)[1]	matherr(GLIBC_2.0)[1]
atan2l(GLIBC_2.0)[1]	erfcf(GLIBC_2.0)[1]	modf(GLIBC_2.0)[1]
atanf(GLIBC_2.0)[1]	erfcl(GLIBC_2.0)[1]	modff(GLIBC_2.0)[1]
atanh(GLIBC_2.0)[1]	erff(GLIBC_2.0)[1]	modfl(GLIBC_2.0)[1]
atanhf(GLIBC_2.0)[1]	erfl(GLIBC_2.0)[1]	nan(GLIBC_2.0)[1]
atanhl(GLIBC_2.0)[1]	exp(GLIBC_2.0)[1]	nanf(GLIBC_2.0)[1]
atanl(GLIBC_2.0)[1]	expf[1]	nanl(GLIBC_2.0)[1]
cabs(GLIBC_2.1)[1]	expl[1]	nearbyint(GLIBC_2.1)[1]
cabsf(GLIBC_2.1)[1]	expm1(GLIBC_2.1)[1]	nearbyintf(GLIBC_2.1)[1]
cabsl(GLIBC_2.1)[1]	fabs(GLIBC_2.1)[1]	nearbyintl(GLIBC_2.1)[1]
cacos(GLIBC_2.1)[1]	fabsf(GLIBC_2.1)[1]	nextafter(GLIBC_2.1)[1]
cacosf(GLIBC_2.1)[1]	fabsl(GLIBC_2.1)[1]	nextafterf(GLIBC_2.1)[1]
cacosh(GLIBC_2.1)[1]	fdim(GLIBC_2.1)[1]	nextafterl(GLIBC_2.1)[1]
cacoshf(GLIBC_2.1)[1]	fdimf(GLIBC_2.1)[1]	nexttoward(GLIBC_2.1)[1]
cacoshl(GLIBC_2.1)[1]	fdiml(GLIBC_2.1)[1]	nexttowardf(GLIBC_2.1)[1]
cacosl(GLIBC_2.1)[1]	feclearexcept(GLIBC_2.1)[1]	nexttowardl(GLIBC_2.1)[1]
carg(GLIBC_2.1)[1]	fegetenv(GLIBC_2.1)[1]	pow(GLIBC_2.1)[1]
cargf(GLIBC_2.1)[1]	fegetexceptflag(GLIBC_2.1)[1]	pow10(GLIBC_2.1)[1]
cargl(GLIBC_2.1)[1]	fegetround(GLIBC_2.1)[1]	pow10f(GLIBC_2.1)[1]
casin(GLIBC_2.1)[1]	feholdexcept(GLIBC_2.1)[1]	pow10l(GLIBC_2.1)[1]
casinf(GLIBC_2.1)[1]	feraiseexcept(GLIBC_2.1)[1]	powf(GLIBC_2.1)[1]
casinh(GLIBC_2.1)[1]	fesetenv(GLIBC_2.1)[1]	powl(GLIBC_2.1)[1]
casinhf(GLIBC_2.1)[1]	fesetexceptflag(GLIBC_2.1)[1]	remainder(GLIBC_2.1)[1]

casinhl(GLIBC_2.1)[1]	fesetround(GLIBC_2.1)[1]	remainderf(GLIBC_2.1)[1]
casinl(GLIBC_2.1)[1]	fetestexcept(GLIBC_2.1)[1]	remainderl(GLIBC_2.1)[1]
catan(GLIBC_2.1)[1]	feupdateenv(GLIBC_2.1)[1]	remquo(GLIBC_2.1)[1]
catanf(GLIBC_2.1)[1]	finite(GLIBC_2.1)[1]	remquof(GLIBC_2.1)[1]
catanh(GLIBC_2.1)[1]	finitef(GLIBC_2.1)[1]	remquol(GLIBC_2.1)[1]
catanhf(GLIBC_2.1)[1]	finitel(GLIBC_2.1)[1]	rint(GLIBC_2.1)[1]
catanhl(GLIBC_2.1)[1]	floor(GLIBC_2.1)[1]	rintf(GLIBC_2.1)[1]
catanl(GLIBC_2.1)[1]	floorf(GLIBC_2.1)[1]	rintl(GLIBC_2.1)[1]
cbrt(GLIBC_2.0)[1]	floorl(GLIBC_2.0)[1]	round(GLIBC_2.0)[1]
cbrtf(GLIBC_2.0)[1]	fma(GLIBC_2.0)[1]	roundf(GLIBC_2.0)[1]
cbrtl(GLIBC_2.0)[1]	fmaf(GLIBC_2.0)[1]	roundl(GLIBC_2.0)[1]
ccos(GLIBC_2.1)[1]	fmal(GLIBC_2.1)[1]	scalb(GLIBC_2.1)[1]
ccosf(GLIBC_2.1)[1]	fmax(GLIBC_2.1)[1]	scalbf(GLIBC_2.1)[1]
ccosh(GLIBC_2.1)[1]	fmaxf(GLIBC_2.1)[1]	scalbl(GLIBC_2.1)[1]
ccoshf(GLIBC_2.1)[1]	fmaxl(GLIBC_2.1)[1]	scalbln(GLIBC_2.1)[1]
ccoshl(GLIBC_2.1)[1]	fmin(GLIBC_2.1)[1]	scalblnf(GLIBC_2.1)[1]
ccosl(GLIBC_2.1)[1]	fminf(GLIBC_2.1)[1]	scalblnl(GLIBC_2.1)[1]
ceil(GLIBC_2.0)[1]	fminl(GLIBC_2.0)[1]	scalbn(GLIBC_2.0)[1]
ceilf(GLIBC_2.0)[1]	fmod(GLIBC_2.0)[1]	scalbnf(GLIBC_2.0)[1]
ceill(GLIBC_2.0)[1]	fmodf(GLIBC_2.0)[1]	scalbnl(GLIBC_2.0)[1]
cexp(GLIBC_2.1)[1]	fmodl(GLIBC_2.1)[1]	significand(GLIBC_2.1)[1]
cexpf(GLIBC_2.1)[1]	frexp(GLIBC_2.1)[1]	significandf(GLIBC_2.1)[1]
cexpl(GLIBC_2.1)[1]	frexpf(GLIBC_2.1)[1]	significandl(GLIBC_2.1)[1]
cimag(GLIBC_2.1)[1]	frexpl(GLIBC_2.1)[1]	sin(GLIBC_2.1)[1]
cimagf(GLIBC_2.1)[1]	gamma(GLIBC_2.1)[1]	sincos(GLIBC_2.1)[1]
cimagl(GLIBC_2.1)[1]	gammaf(GLIBC_2.1)[1]	sincosf(GLIBC_2.1)[1]
clog(GLIBC_2.1)[1]	gammal(GLIBC_2.1)[1]	sincosl(GLIBC_2.1)[1]
clog10(GLIBC_2.1)[1]	hypot(GLIBC_2.1)[1]	sinf(GLIBC_2.1)[1]
clog10f(GLIBC_2.1)[1]	hypotf(GLIBC_2.1)[1]	sinh(GLIBC_2.1)[1]
clog10l(GLIBC_2.1)[1]	hypotl(GLIBC_2.1)[1]	sinhf(GLIBC_2.1)[1]
clogf(GLIBC_2.1)[1]	ilogb(GLIBC_2.1)[1]	sinhl(GLIBC_2.1)[1]

clogl(GLIBC_2.1)[1]	ilogbf(GLIBC_2.1)[1]	sinl(GLIBC_2.1)[1]
conj(GLIBC_2.1)[1]	ilogbl(GLIBC_2.1)[1]	sqrt(GLIBC_2.1)[1]
conjf(GLIBC_2.1)[1]	j0(GLIBC_2.1)[1]	sqrtf(GLIBC_2.1)[1]
conjl(GLIBC_2.1)[1]	j0f(GLIBC_2.1)[1]	sqrtl(GLIBC_2.1)[1]
copysign(GLIBC_2.0)[1]	j0l(GLIBC_2.0)[1]	tan(GLIBC_2.0)[1]
copysignf(GLIBC_2.0)[1]	j1(GLIBC_2.0)[1]	tanf(GLIBC_2.0)[1]
copysignl(GLIBC_2.0)[1]	j1f(GLIBC_2.0)[1]	tanh(GLIBC_2.0)[1]
cos(GLIBC_2.0)[1]	j1l(GLIBC_2.0)[1]	tanhf(GLIBC_2.0)[1]
cosf(GLIBC_2.0)[1]	jn(GLIBC_2.0)[1]	tanhl(GLIBC_2.0)[1]
cosh(GLIBC_2.0)[1]	jnf(GLIBC_2.0)[1]	tanl(GLIBC_2.0)[1]
coshf(GLIBC_2.0)[1]	jnl(GLIBC_2.0)[1]	tgamma(GLIBC_2.0)[1]
coshl(GLIBC_2.0)[1]	ldexp(GLIBC_2.0)[1]	tgammaf(GLIBC_2.0)[1]
cosl(GLIBC_2.0)[1]	ldexpf(GLIBC_2.0)[1]	tgammal(GLIBC_2.0)[1]
cpow(GLIBC_2.1)[1]	ldexpl(GLIBC_2.1)[1]	trunc(GLIBC_2.1)[1]
cpowf(GLIBC_2.1)[1]	lgamma(GLIBC_2.1)[1]	truncf(GLIBC_2.1)[1]
cpowl(GLIBC_2.1)[1]	lgamma_r(GLIBC_2.1)[1]	truncl(GLIBC_2.1)[1]
cproj(GLIBC_2.1)[1]	lgammaf(GLIBC_2.1)[1]	y0(GLIBC_2.1)[1]
cprojf(GLIBC_2.1)[1]	lgammaf_r(GLIBC_2.1)[1]	y0f(GLIBC_2.1)[1]
cprojl(GLIBC_2.1)[1]	lgammal(GLIBC_2.1)[1]	y0l(GLIBC_2.1)[1]
creal(GLIBC_2.1)[1]	lgammal_r(GLIBC_2.1)[1]	y1(GLIBC_2.1)[1]
crealf(GLIBC_2.1)[1]	llrint(GLIBC_2.1)[1]	y1f(GLIBC_2.1)[1]
creall(GLIBC_2.1)[1]	llrintf(GLIBC_2.1)[1]	y1l(GLIBC_2.1)[1]
csin(GLIBC_2.1)[1]	llrintl(GLIBC_2.1)[1]	yn(GLIBC_2.1)[1]
csinf(GLIBC_2.1)[1]	llround(GLIBC_2.1)[1]	ynf(GLIBC_2.1)[1]
csinh(GLIBC_2.1)[1]	llroundf(GLIBC_2.1)[1]	ynl(GLIBC_2.1)[1]
csinhf(GLIBC_2.1)[1]	llroundl(GLIBC_2.1)[1]	

Table A-6. libm Data Interfaces

signgamID_STD_46_SUSV3		

A.5. libncurses

The behaviour of the interfaces in this library is specified by the following Standards.

X/Open Curses

Table A-7. 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]
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]	syncok[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]	mvwaddch[1]	tigetflag[1]
can_change_color[1]	mvwaddchnstr[1]	tigetnum[1]
cbreak[1]	mvwaddchstr[1]	tigetstr[1]
chgat[1]	mvwaddnstr[1]	timeout[1]
clear[1]	mvwaddstr[1]	touchline[1]
clearok[1]	mvwchgat[1]	touchwin[1]
clrtobot[1]	mvwdelch[1]	tparm[1]
clrtoeol[1]	mvwgetch[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]
delch[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]	waddch[1]
doupdate[1]	mvwscanw[1]	waddchnstr[1]
dupwin[1]	mvwvline[1]	waddchstr[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]	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]	scrl[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]	setscrreg[1]	wscanw[1]
longname[1]	setupterm[1]	wscrl[1]
meta[1]	slk_attr_set[1]	wsetscrreg[1]
move[1]	slk_attroff[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

COLORSID_STD_46_SUS_46_CURSES	LINESID_STD_46_SUS_46_CURSES	curscrID_STD_46_SUS_46_CURSES
COLOR_PAIRSID_STD_46_SUS_46_CURSES	acs_mapID_STD_46_SUS_46_CURSES	stdscrID_STD_46_SUS_46_CURSES
COLSID_STD_46_SUS_46_CURSES	cur_termID_STD_46_SUS_46_CURSES	

A.6. libpam

The behaviour of the interfaces in this library is specified by the following Standards.

this specification

Table A-9. 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_chauthtok[1]	pam_getenvlist[1]	pam_strerror[1]
pam_close_session[1]	pam_open_session[1]	
pam_end[1]	pam_set_item[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

_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]
pthread_attr_init(GLIBC_2.1)[1]	pthread_mutex_init(GLIBC_2.1)[1]	pthread_setconcurrency[1]
pthread_attr_setdetachstate(GLIBC_2.0)[1]	pthread_mutex_lock(GLIBC_2.0)[1]	pthread_setspecific(GLIBC_2.0)[1]
pthread_attr_setguardsize(GLIBC_2.1)[1]	pthread_mutex_trylock(GLIBC_2.1)[1]	pthread_sigmask(GLIBC_2.1)[1]
pthread_attr_setschedparam(GLIBC_2.0)[1]	pthread_mutex_unlock(GLIBC_2.0)[1]	pthread_testcancel(GLIBC_2.0)[1]
pthread_attr_setstackaddr(GLIBC_2.1)[1]	pthread_mutexattr_destroy(GLIBC_2.1)[1]	pwrite(GLIBC_2.1)[1]
pthread_attr_setstacksize(GLIBC_2.1)[1]	pthread_mutexattr_getpshared(GLIBC_2.1)[1]	pwrite64(GLIBC_2.1)[1]
pthread_cancel(GLIBC_2.0)[1]	pthread_mutexattr_gettype(GLIBC_2.0)[1]	sem_close(GLIBC_2.0)[1]
pthread_cond_broadcast(GLIBC_2.0)[1]	pthread_mutexattr_init(GLIBC_2.0)[1]	sem_destroy(GLIBC_2.0)[1]
pthread_cond_destroy(GLIBC_2.0)[1]	pthread_mutexattr_setpshared(GLIBC_2.0)[1]	sem_getvalue(GLIBC_2.0)[1]
pthread_cond_init(GLIBC_2.0)[1]	pthread_mutexattr_settype(GLIBC_2.0)[1]	sem_init(GLIBC_2.0)[1]
pthread_cond_signal(GLIBC_2.0)[1]	pthread_once(GLIBC_2.0)[1]	sem_open(GLIBC_2.0)[1]
pthread_cond_timedwait(GLIBC_2.0)[1]	pthread_rwlock_destroy(GLIBC_2.0)[1]	sem_post(GLIBC_2.0)[1]

pthread_cond_wait(GLIBC_2.0)[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. libutil

The behaviour of the interfaces in this library is specified by the following Standards.

this specification

Table A-11. 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.9. libz

The behaviour of the interfaces in this library is specified by the following Standards.

zlib Manual

Table A-12. libz Function Interfaces

adler32[1]	gzdopen[1]	gztell[1]
compress[1]	gzeof[1]	gzwrite[1]
compress2[1]	gzerror[1]	inflate[1]
crc32[1]	gzflush[1]	inflateEnd[1]
deflate[1]	gzgetc[1]	inflateInit2_[1]
deflateCopy[1]	gzgets[1]	inflateInit_[1]
deflateEnd[1]	gzopen[1]	inflateReset[1]
deflateInit2_[1]	gzprintf[1]	inflateSetDictionary[1]
deflateInit_[1]	gzputc[1]	inflateSync[1]
deflateParams[1]	gzputs[1]	inflateSyncPoint[1]
deflateReset[1]	gzread[1]	uncompress[1]

deflateSetDictionary[1]	gzrewind[1]	zError[1]
get_crc_table[1]	gzseek[1]	
gzclose[1]	gzsetparams[1]	

42

Linux Packaging Specification

Linux Packaging Specification

Table of Contents

List of Tables

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".

major

Value indicating the major version number of the file format version. This value shall be 3.

minor

Value indicating the minor revision number of file format version. This value shall be 0.

type

Value indicating whether this is a source or binary package. This value shall be 0 to indicate a binary package.

archnum

Value indicating the architecture for which this package is valid. This value is specified in the architecture-specific LSB specification.

name

A NUL terminated string that provides the package name. This name shall conform with the Package Naming section of this specification.

osnum

Value indicating the Operating System for which this package is valid. This value shall be 1.

signature_type

Value indicating the type of the signature used in the Signature part of the file. This value shall be 5.

reserved

Reserved space. The value is undefined.

1.1.2. Header Structure

The Header structure is used for both the Signature and Header Sections. A Header Structure consists of 3 parts, a Header record, followed by 1 or more Index records, followed by 0 or more bytes of data associated with the Index records. A Header structure shall be aligned to an 8 byte boundary.

Table 1-2. Signature Format

Header Record
Array of Index Records
Store of Index Values

1.1.2.1. Header Record

```
struct rpmheader {
    unsigned char magic[4];
    unsigned char reserved[4];
    int nindex;
    int hsize;
    } ;
```

magic

Value identifying this record as an RPM header record. This value shall be "\216\255\350\001".

reserved

Reserved space. This value shall be "\000\000\000\000".

nindex

The number of Index Records that follow this Header Record. There should be at least 1 Index Record.

hsize

The size in bytes of the storage area for the data pointed to by the Index Records.

1.1.2.2. Index Record

```
struct rpmhdrindex {
    int tag;
    int type;
    int offset;
    int count;
    } ;
```

tag

Value identifying the purpose of the data associated with this Index Record. This value of this field is dependent on the context in which the Index Record is used, and is defined below and in later sections.

type

Value identifying the type of the data associated with this Index Record. The possible *type* values are defined below.

offset

Location in the Store of the data associated with this Index Record. This value should between 0 and the value contained in the *hsize* of the Header Structure.

count

Size of the data associated with this Index Record. The *count* is the number of elements whose size is defined by the type of this Record.

1.1.2.2.1. Index Type Values

The possible values for the *type* field are defined in this table.

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 enties 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_HDRI18NTABLE` 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_HEADERI18NTABLE	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_HEADERI18NTABLE

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

153 The Header section is implemented using the Header structure. The Header section defines the following additional
154 tag values which may be used in the Index structures.

155 1.1.4.1. Package Information

156 The following tag values are used to indicate information that describes the package as a whole.

157 **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_SOURCERPM	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_PAYLOADFLAGS	1126	STRING	1	Required

RPMTAG_NAME

This tag specifies the name of the package.

RPMTAG_VERSION

This tag specifies the version of the package.

RPMTAG_RELEASE

This tag specifies the release of the package.

RPMTAG_SUMMARY

This tag specifies the summary description of the package. The summary value pointed to by this index record contains a one line description of the package.

RPMTAG_DESCRIPTION

This tag specifies the description of the package. The description value pointed to by this index record contains a full desription of the package.

RPMTAG_SIZE

This tag specifies the sum of the sizes of the regular files in the archive.

RPMTAG_LICENSE

This tag specifies the license which applies to this package.

RPMTAG_GROUP

This tag specifies the administrative group to which this package belongs.

RPMTAG_OS

This tag specifies the OS of the package. The OS value pointed to by this index record shall be "linux".

RPMTAG_ARCH

This tag specifies the architecture of the package. The architecture value pointed to by this index record is defined in architecture specific LSB specification.

RPMTAG_SOURCERPM

This tag specifies the name of the source RPM

RPMTAG_ARCHIVESIZE

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.

RPMTAG_PREUN

his tag specifies the preuninstall scriptlet.

RPMTAG_POSTUN

This tag specified the postuninstall scriptlet.

RPMTAG_PREINPROG

This tag specifies the name of the intepreter to which the preinstall scriptlet will be passed. The intepreter pointed to by this index record shall be '/bin/sh'.

RPMTAG_POSTINPROG

This tag specifies the name of the intepreter to which the postinstall scriptlet will be passed. The intepreter pointed to by this index record shall be '/bin/sh'.

RPMTAG_PREUNPROG

This tag specifies the name of the intepreter to which the preuninstall scriptlet will be passed. The intepreter pointed to by this index record shall be '/bin/sh'.

RPMTAG_POSTUNPROG

This program specifies the name of the intepreter to which the postuninstall scriptlet will be passed. The intepreter pointed to by this index record shall be '/bin/sh'.

1.1.4.3. File Information

The following tag values are used to provide information about the files in the payload. This information is provided in the header to allow more efficient access of the information.

Table 1-10. File Info Tag Values

Name	Tag Value	Type	Count	Status
RPMTAG_OLDFILENAMES	1027	STRING_ARRAY		Optional
RPMTAG_FILESIZES	1028	INT32		Required
RPMTAG_FILEMODES	1030	INT16		Required
RPMTAG_FILERDEVS	1033	INT16		Required
RPMTAG_FILEMTIMES	1034	INT32		Required
RPMTAG_FILEMD5S	1035	STRING_ARRAY		Required
RPMTAG_FILELI	1036	STRING_ARRAY		Required

Name	Tag Value	Type	Count	Status
NKTOS				
RPMTAG_FILEFLAGS	1037	INT32		Required
RPMTAG_FILEUSERNAME	1039	STRING_ARRAY		Required
RPMTAG_FILEGROUPNAME	1040	STRING_ARRAY		Required
RPMTAG_FILEDEVICES	1095	INT32		Required
RPMTAG_FILEINODES	1096	INT32		Required
RPMTAG_FILELANGS	1097	STRING_ARRAY		Required
RPMTAG_DIRINDEXES	1116	INT32		Optional
RPMTAG_BASENAMES	1117	STRING_ARRAY		Optional
RPMTAG_DIRNAMES	1118	STRING_ARRAY		Optional

RPMTAG_OLDFILENAMES

This tag specifies the filenames when not in a compressed format as determined by the absense of rpmlib(CompressedFileNames) in the RPMTAG_REQUIRENAME index.

RPMTAG_FILESIZES

This tag specifies the size of each file in the archive.

RPMTAG_FILEMODES

This tag specifies the mode of each file in the archive.

RPMTAG_FILERDEVS

This tag specifies the device number from which the file was copied.

RPMTAG_FILEMTIMES

This tag specifies the modification time in seconds since the epoch of each file in the archive.

RPMTAG_FILEMD5S

This tag specifies the ASCII representation of the MD5 sum of the corresponding file contents. This value is empty if the corresponding archive entry is not a regular file.

RPMTAG_FILELINKTOS

The target for a symlink, otherwise NULL.

RPMTAG_FILEFLAGS

This tag specifies the bit(s) to classify and control how files are to be installed.

RPMTAG_FILEUSERNAME

This tag specifies the owner of the corresponding file.

RPMTAG_FILEGROUPNAME

This tag specifies the of the corresponding file.

RPMTAG_FILEDEVICES

This tag specifies the 16 bit device number from which the file was copied.

RPMTAG_FILEINODES

This tag specifies the inode value from the original file on the build host.

RPMTAG_FILELANGS

This tag specifies a per-file locale marker used to install only locale specific subsets of files when the package is installed.

RPMTAG_DIRINDEXES

This tag specifies the index into the array provided by the RPMTAG_DIRNAMES Index which contains the directory name for the corresponding filename.

RPMTAG_BASENAMES

This tag specifies the base portion of the corresponding filename.

RPMTAG_DIRNAMES

This tag specifies the directory portion of the corresponding filename. Each directory name shall contain a trailing '/'.

One of RPMTAG_OLDFILENAMES or the tuple
RPMTAG_DIRINDEXES,RPMTAG_BASENAMES,RPMTAG_DIRNAMES shall be present, but not both.

1.1.4.4. Dependency Information

The following tag values are used to provide information about interdependencies between packages.

Table 1-11. Package Dependency Tag Values

Name	Tag Value	Type	Count	Status
RPMTAG_PROVIDENAME	1047	STRING_ARRAY	1	Required
RPMTAG_REQUI	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

RPMTAG_PROVIDENAME

This tag indicates the name of the dependency provided by this package.

RPMTAG_REQUIREFLAGS

Bits(s) to specify the dependency range and context.

RPMTAG_REQUIRENAME

This tag indicates the dependencies for this package.

RPMTAG_REQUIREVERSION

This tag indicates the versions associated with the values found in the RPMTAG_REQUIRENAME Index.

RPMTAG_CONFLICTFLAGS

Bits(s) to specify the conflict range and context.

RPMTAG_CONFLICTNAME

This tag indicates the conflictind dependencies for this package.

RPMTAG_CONFLICTVERSION

This tag indicates the versions associated with the values found in the RPMTAG_CONFLICTNAME Index.

RPMTAG_OBSOLETENAME

This tag indicates the obsoleted dependencies for this package.

RPMTAG_PROVIDEFLAGS

Bits(s) to specify the conflict range and context.

RPMTAG_PROVIDEVERSION

This tag indicates the versions associated with the values found in the RPMTAG_PROVIDENAME Index.

RPMTAG_OBSOLETEFLAGS

Bits(s) to specify the conflict range and context.

RPMTAG_OBSOLETEVERSION

This tag indicates the versions associated with the values found in the RPMTAG_OBSOLETENAME Index.

1.1.4.4.1. Package Dependency Values

The package dependencies are stored in the `RPMTAG_REQUIRENAME` and `RPMTAG_REQUIREVERSION` index records. The following values may be used.

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_OBSOLETENAME 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_BASENAMES 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
RPMSENSE_LESS	0x02	
RPMSENSE_GREATER	0x04	
RPMSENSE_EQUAL	0x08	
RPMSENSE_PREREQ	0x40	
RPMSENSE_INTERP	0x100	
RPMSENSE_SCRIPT_PRE	0x200	
RPMSENSE_SCRIPT_POST	0x400	
RPMSENSE_SCRIPT_PREUN	0x800	
RPMSENSE_SCRIPT_POSTUN	0x1000	
RPMSENSE_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_BUILDTIME	1006	INT32	1	Optional
RPMTAG_BUILDHOST	1007	STRING	1	Optional
RPMTAG_FILEVERIFYFLAGS	1045	INT32		Optional
RPMTAG_CHANG	1080	INT32		Optional

Name	Tag Value	Type	Count	Status
ELOGTIME				
RPMTAG_CHANGELOGNAME	1081	STRING_ARRAY		Optional
RPMTAG_CHANGELOGTEXT	1082	STRING_ARRAY		Optional
RPMTAG_OPTFLAGS	1122	STRING	1	Optional
RPMTAG_RHNPLATFORM	1131	STRING	1	Deprecated
RPMTAG_PLATFORM	1132	STRING	1	Optional

RPMTAG_BUILDTIME

This tag specifies the time as seconds since the epoch at which the package was built.

RPMTAG_BUILDHOST

This tag specifies the on which which the package was built.

RPMTAG_FILEVERIFYFLAGS

This tag specifies the bit(s) to control how files are to be verified after install, specifying which checks should be performed.

RPMTAG_CHANGELOGTIME

This tag specifies the Unix time in seconds since the epoch associated with each entry in the Changelog file.

RPMTAG_CHANGELOGNAME

This tag specifies the name of who made a change to this package

RPMTAG_CHANGELOGTEXT

This tag specifies the changes asssociated with a changelog entry.

RPMTAG_OPTFLAGS

This tag indicates additional flags which may have been passed to the compiler when building this package.

RPMTAG_RHNPLATFORM

This tag contains an opaque string whose contents are undefined.

RPMTAG_PLATFORM

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.

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".

c_ino

This field contains the inode number from the filesystem from which the file was read. This field is ignored when installing a package. This field shall match the corresponding value in the `RPMTAG_FILEINODES` index in the Header section.

c_mode

Permission bits of the file. This is an ascii representation of the hexadecimal number representing the bit as defined for the *st_mode* field of the stat structure defined for the `stat` function. This field shall match the corresponding value in the `RPMTAG_FILEMODES` index in the Header section.

c_uid

Value identifying this owner of this file. This value matches the uid value of the corresponding user in the RPMTAG_FILEUSERNAME as found on the system where this package was built. The username specified in RPMTAG_FILEUSERNAME should take precedence when installing the package.

c_gid

Value identifying this group of this file. This value matches the gid value of the corresponding user in the RPMTAG_FILEGROUPNAME as found on the system where this package was built. The groupname specified in RPMTAG_FILEGROUPNAME should take precedence when installing the package.

c_nlink

Value identifying the number of links associated with this file. If the value is greater than 1, then this filename will be linked to 1 or more files in this archive that has a matching value for the c_ino, c_devmajor and c_devminor fields.

c_mtime

Value identifying the modification time of the file when it was read. This field shall match the corresponding value in the `RPMTAG_FILEMTIMES` index in the Header section.

c_filesize

Value identifying the size of the file. This field shall match the corresponding value in the `RPMTAG_FILESIZES` index in the Header section.

c_devmajor

The major number of the device containing the file system from which the file was read. With the exception of processing files with c_nlink >1, this field is ignored when installing a package. This field shall match the corresponding value in the `RPMTAG_FILEDEVICES` index in the Header section.

c_devminor

The minor number of the device containing the file system from which the file was read. With the exception of processing files with c_nlink >1, this field is ignored when installing a package. This field shall match the 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 seperated 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.

2. The distribution itself may use a different packaging format for its own packages, and of course it may use any available mechanism for installing the LSB-conformant packages.
3. For example, there are discrepancies among distributions concerning whether the name might be frobnicator-1.7-21-ppc32.rpm or frobnicator-1.7-21-powerpc32.rpm. The architecture aside, recommended practice is for the filename of the package file to match the name within the package.
4. For example, if the name with the release and version is frobnicator-1.7-21, the name part is frobnicator and falls under the rules for a name with no hyphens.
5. For example, "frobnicator".
6. For example, ssh-common, ssh-client, kernel-pcmcia, and the like. Possible alternative names include sshcommon, foolinux-ssh-common (where foolinux is registered to the distribution), or lsb-foolinux-ssh-common.
7. For example, if an application vendor has domain name visicalc.example.com and has registered visicalc as a provider name, they might name packages visicalc-base, visicalc.example.com-charting, and the like.

Free Documentation License

Free Documentation License

Table of Contents

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

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. Secondarily, 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.