Linux Standard Base Core Specification 3.2
## Contents

### I Introductory Elements  
1 Scope........................................................................................................1  
1.1 General..............................................................................................1  
1.2 Module Specific Scope.................................................................1  
2 References.............................................................................................2  
2.1 Normative References ....................................................................2  
2.2 Informative References/Bibliography .....................................3  
3 Requirements..........................................................................................6  
3.1 Relevant Libraries ........................................................................6  
3.2 LSB Implementation Conformance ......................................6  
3.3 LSB Application Conformance ...............................................7  
4 Definitions............................................................................................9  
5 Terminology..........................................................................................10  
6 Documentation Conventions ................................................................12  
7 Relationship To ISO/IEC 9945 POSIX ...........................................13  
8 Relationship To Other Linux Foundation Specifications ..............14  

### II Executable And Linking Format (ELF)  
9 Introduction.........................................................................................16  
10 Low Level System Information.....................................................17  
10.1 Operating System Interface .....................................................17  
10.2 Machine Interface .......................................................................17  
11 Object Format.....................................................................................18  
11.1 Object Files ..................................................................................18  
11.2 Sections ......................................................................................18  
11.3 Special Sections..........................................................................22  
11.4 Symbol Mapping .........................................................................27  
11.5 DWARF Extensions .....................................................................27  
11.6 Exception Frames ..........................................................................31  
11.7 Symbol Versioning .........................................................................36  
11.8 ABI note tag ................................................................................39  
12 Dynamic Linking..................................................................................41  
12.1 Program Loading and Dynamic Linking ................................41  
12.2 Program Header ...........................................................................41  
12.3 Dynamic Entries ...........................................................................41  

### III Base Libraries  
13 Base Libraries ......................................................................................47  
13.1 Introduction ...................................................................................47  
13.2 Program Interpreter .................................................................47  
13.3 Interfaces for libc .........................................................................47  
13.4 Data Definitions for libc .............................................................62  
13.5 Interface Definitions for libc ......................................................120  
13.6 Interfaces for libm ...............................................................241  
13.7 Data Definitions for libm ...........................................................244  
13.8 Interface Definitions for libm ......................................................251  
13.9 Interfaces for libpthread ...........................................................273  
13.10 Data Definitions for libpthread .............................................276  
13.11 Interface Definitions for libpthread ..................................281  
13.12 Interfaces for libgcc_s .............................................................282  
13.13 Data Definitions for libgcc_s .................................................282  
13.14 Interfaces for libdl ...............................................................283  
13.15 Data Definitions for libdl .......................................................284  

© 2007 Linux Foundation
List of Figures

11-1 Version Definition Entries ................................................................. 36
11-2 Version Definition Auxiliary Entries .................................................. 37
11-3 Version Needed Entries ..................................................................... 37
11-4 Version Needed Auxiliary Entries .................................................... 38
12-1 Dynamic Structure ............................................................................ 41
Foreword

This is version 3.2 of the Linux Standard Base Core Specification. This specification is part of a family of specifications under the general title "Linux Standard Base". Developers of applications or implementations interested in using the LSB trademark should see the Linux Foundation Certification Policy for details.
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.

The specification carries a version number of either the form $x.y$ or $x.y.z$. This version number carries the following meaning:

- The first number ($x$) is the major version number. All versions with the same major version number should share binary compatibility. Any addition or deletion of a new library results in a new version number. Interfaces marked as deprecated may be removed from the specification at a major version change.

- The second number ($y$) is the minor version number. Individual interfaces may be added if all certified implementations already had that (previously undocumented) interface. Interfaces may be marked as deprecated at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.

- The third number ($z$), if present, is the editorial level. Only editorial changes should be included in such versions.

Since this specification is a descriptive Application Binary Interface, and not a source level API specification, it is not possible to make a guarantee of 100% backward compatibility between major releases. However, it is the intent that those parts of the binary interface that are visible in the source level API will remain backward compatible from version to version, except where a feature marked as "Deprecated" in one release may be removed from a future release. Implementors are strongly encouraged to make use of symbol versioning to permit simultaneous support of applications conforming to different releases of this specification.
I Introductory Elements
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" or "generic LSB"), ISO/IEC 23360 Part 1, describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific part ("LSB-arch" or "archLSB") describing the parts of the interface that vary by processor architecture. Together, the LSB-generic and the relevant architecture-specific part of ISO/IEC 23360 for a single hardware architecture provide a complete interface specification for compiled application programs on systems that share a common hardware architecture.

ISO/IEC 23360 Part 1, the LSB-generic document, should be used in conjunction with an architecture-specific part. Whenever a section of the LSB-generic specification is supplemented by architecture-specific information, the LSB-generic document includes a reference to the architecture part. Architecture-specific parts of ISO/IEC 23360 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 provides 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 Standard Base (LSB), ISO/IEC 23360 Part 1. This module provides the fundamental system interfaces, libraries, and runtime environment upon which all conforming applications and libraries depend.

Interfaces described in this part of ISO/IEC 23360 are mandatory except where explicitly listed otherwise. Core interfaces may be supplemented by other modules; all modules are built upon the core.
2 References

2.1 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Note: Where copies of a document are available on the World Wide Web, a Uniform Resource Locator (URL) is given for informative purposes only. This may point to a more recent copy of the referenced specification, or may be out of date. Reference copies of specifications at the revision level indicated may be found at the Linux Foundation’s Reference Specifications (http://refspecs.freestandards.org) site.

Table 2-1 Normative References

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filesystem Hierarchy Standard</td>
<td>Filesystem Hierarchy Standard (FHS) 2.3</td>
<td><a href="http://www.pathname.com/fhs/">http://www.pathname.com/fhs/</a></td>
</tr>
<tr>
<td></td>
<td>ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Including Technical Cor. 1: 2004</td>
<td></td>
</tr>
</tbody>
</table>
### References

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itanium™ C++ ABI</td>
<td>Itanium™ C++ ABI (Revision 1.83)</td>
<td><a href="http://refspecs.linux-foundation.org/cxxabi-1.83.html">http://refspecs.linux-foundation.org/cxxabi-1.83.html</a></td>
</tr>
<tr>
<td>SVID Issue 4</td>
<td>System V Interface Definition, Fourth Edition</td>
<td></td>
</tr>
</tbody>
</table>

### 2.2 Informative References/Bibliography

In addition, the specifications listed below provide essential background information to implementors of this specification. These references are included for information only.

**Table 2-2 Other References**

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWARF Debugging Information Format, Revision 2.0.0</td>
<td>DWARF Debugging Information Format, Revision 2.0.0 (July 27,</td>
<td><a href="http://refspecs.linux-foundation.org/dwarf/dwarf-2.0.0.pdf">http://refspecs.linux-foundation.org/dwarf/dwarf-2.0.0.pdf</a></td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td>URL</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>Linux Allocated Device Registry</td>
<td>LINUX ALLOCATED DEVICES</td>
<td><a href="http://www.lanana.org/docs/device-list/devices.txt">http://www.lanana.org/docs/device-list/devices.txt</a></td>
</tr>
<tr>
<td>PAM</td>
<td>Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar &amp; R.Schemers (SunSoft)</td>
<td><a href="http://www.opengroup.org/tech/rfc/mirror/rfc/rfc86.0.txt">http://www.opengroup.org/tech/rfc/mirror/rfc/rfc86.0.txt</a></td>
</tr>
<tr>
<td>RFC 1831/1832 RPC &amp; XDR</td>
<td>IETF RFC 1831 &amp; 1832</td>
<td><a href="http://www.ietf.org/">http://www.ietf.org/</a></td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td>URL</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Format Specification</td>
<td>Data Format Specification version 1.3</td>
<td></td>
</tr>
</tbody>
</table>
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 relevant architecture specific part of ISO/IEC 23360.

Table 3-1 Standard Library Names

<table>
<thead>
<tr>
<th>Library</th>
<th>Runtime Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>libdl</td>
<td>libdl.so.2</td>
</tr>
<tr>
<td>libcrypt</td>
<td>libcrypt.so.1</td>
</tr>
<tr>
<td>libz</td>
<td>libz.so.1</td>
</tr>
<tr>
<td>libncurses</td>
<td>libncurses.so.5</td>
</tr>
<tr>
<td>libutil</td>
<td>libutil.so.1</td>
</tr>
<tr>
<td>libpthread</td>
<td>libpthread.so.0</td>
</tr>
<tr>
<td>librt</td>
<td>librt.so.1</td>
</tr>
<tr>
<td>libpam</td>
<td>libpam.so.0</td>
</tr>
<tr>
<td>libgcc_s</td>
<td>libgcc_s.so.1</td>
</tr>
</tbody>
</table>

Table 3-2 Standard Library Names defined in the Architecture Specific Parts of ISO/IEC 23360

<table>
<thead>
<tr>
<th>Library</th>
<th>Runtime Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>libm</td>
<td>See archLSB</td>
</tr>
<tr>
<td>libc</td>
<td>See archLSB</td>
</tr>
<tr>
<td>proginterp</td>
<td>See archLSB</td>
</tr>
</tbody>
</table>

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 is necessarily architecture specific, and must provide the interfaces specified by both the generic LSB Core specification (ISO/IEC 23360 Part 1) and the relevant architecture specific part of ISO/IEC 23360.

**Rationale:** An implementation must provide at least the interfaces specified in these specifications. It may also provide additional interfaces.

A conforming implementation shall satisfy the following requirements:

- A processor architecture represents a family of related processors which may not have identical feature sets. The architecture specific parts of ISO/IEC 23360 that supplement this specification for a given target processor
architecture describe a minimum acceptable processor. The implementation shall provide all features of this processor, whether in hardware or through emulation transparent to the application.

• 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 is necessarily architecture specific, and must conform to both the generic LSB Core specification (ISO/IEC 23360 Part 1) and the relevant architecture specific part of ISO/IEC 23360.

A conforming application shall satisfy the following requirements:

• Its executable files shall be either shell scripts or object files in the format defined for the Object File Format system interface.

• Its object files shall participate in dynamic linking as defined in the Program Loading and Linking System interface.

• It shall employ 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 shall be stated in the application’s documentation.

• It shall 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 shall be in turn an LSB conforming application.
  • The use of that interface or data format, as well as its source, shall be 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 shall 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.
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 possibility 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
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).
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) [SUSv3]

refers to the interface named forkpty() with symbol version GLIBC_2.0 that is defined in the SUSv3 reference.

Note: Symbol versions are defined in the architecture specific parts of ISO/IEC 23360 only.
7 Relationship To ISO/IEC 9945 POSIX

This specification includes many interfaces described in ISO POSIX (2003). Unless otherwise specified, such interfaces should behave exactly as described in that specification. Any conflict between the requirements described here and the ISO POSIX (2003) standard is unintentional, except as explicitly noted otherwise.

Note: In addition to the differences noted inline in this specification, PDTR 24715 has extracted the differences between this specification and ISO POSIX (2003) into a single place. It is the long term plan of the Linux Foundation to converge the LSB Core Specification with ISO/IEC 9945 POSIX.

The LSB Specification Authority is responsible for deciding the meaning of conformance to normative referenced standards in the LSB context. Problem Reports regarding underlying or referenced standards in any other context will be referred to the relevant maintenance body for that standard.
8 Relationship To Other Linux Foundation Specifications

The LSB is the base for several other specification projects under the umbrella of the Linux Foundation (LF). This specification is the foundation, and other specifications build on the interfaces defined here. However, beyond those specifications listed as Normative References, this specification has no dependencies on other LF projects.
II Executable And Linking Format (ELF)
9 Introduction

Executable and Linking Format (ELF) defines the object format for compiled applications. This specification supplements the information found in System V ABI Update and is intended to document additions made since the publication of that document.
10 Low Level System Information

10.1 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.

10.2 Machine Interface

10.2.1 Data Representation

LSB-conforming applications shall use the data representation as defined in the Architecture specific ELF documents.

10.2.1.1 Fundamental Types

In addition to the fundamental types specified in the relevant architecture specific part of ISO/IEC 23360, a 1 byte data type is defined here.

Table 10-1 Scalar Types

<table>
<thead>
<tr>
<th>Type</th>
<th>C</th>
<th>C++</th>
<th>sizeof</th>
<th>Alignment (bytes)</th>
<th>Architecture Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral</td>
<td>_Bool</td>
<td>bool</td>
<td>1</td>
<td>1</td>
<td>byte</td>
</tr>
</tbody>
</table>
11 Object Format

11.1 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 specification
- the relevant architecture specific part of ISO/IEC 23360

Conforming implementations may also support other unspecified object file formats.

11.2 Sections

11.2.1 Introduction

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

11.2.2 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 11-1, drawn from the System V ABI, or one of the additional values specified in Table 11-2.

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

11.2.2.1 ELF Section Types

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

Table 11-1 ELF Section Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHT_DYNAMIC</td>
<td>0x6</td>
<td>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 of System V ABI Update for details.</td>
</tr>
<tr>
<td>SHT_DYNSYM</td>
<td>0xb</td>
<td>This section holds a minimal set of symbols adequate for dynamic linking. See also SHT_SYMTAB. Currently, an object file</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>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.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHT_FINI_ARRAY</td>
<td>0xf</td>
<td>This section contains an array of pointers to termination functions, as described in 'Initialization and Termination Functions' in Chapter 5 of System V ABI Update. Each pointer in the array is taken as a parameterless procedure with a void return.</td>
</tr>
<tr>
<td>SHT_HASH</td>
<td>0x5</td>
<td>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 future. See 'Hash Table' in Chapter 5 of System V ABI Update for details.</td>
</tr>
<tr>
<td>SHT_INIT_ARRAY</td>
<td>0xe</td>
<td>This section contains an array of pointers to initialization functions, as described in 'Initialization and Termination Functions' in Chapter 5 of System V ABI Update. Each pointer in the array is taken as a parameterless procedure with a void return.</td>
</tr>
<tr>
<td>SHT_NOBITS</td>
<td>0x8</td>
<td>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.</td>
</tr>
<tr>
<td>SHT_NOTE</td>
<td>0x7</td>
<td>The section holds in-</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SHT_NULL</td>
<td>0x0</td>
<td>This value marks the section header as inactive; it does not have an associated section. Other members of the section header have undefined values.</td>
</tr>
<tr>
<td>SHT_PREINIT_ARRAY</td>
<td>0x10</td>
<td>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 of System V ABI Update. Each pointer in the array is taken as a parameterless procedure with a void return.</td>
</tr>
<tr>
<td>SHT_PROGBITS</td>
<td>0x1</td>
<td>The section holds information defined by the program, whose format and meaning are determined solely by the program.</td>
</tr>
<tr>
<td>SHT_REL</td>
<td>0x9</td>
<td>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' in Chapter 4 of System V ABI Update for details.</td>
</tr>
<tr>
<td>SHT_RELA</td>
<td>0x4</td>
<td>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</td>
</tr>
</tbody>
</table>
for the 64-bit class of object files. An object file may have multiple relocation sections. See `Relocation' in Chapter 4 of System V ABI Update for details.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHT_STRTAB</td>
<td>0x3</td>
<td>The section holds a string table. An object file may have multiple string table sections. See `String Table' in Chapter 4 of System V ABI Update for details.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHT_SYMTAB</td>
<td>0x2</td>
<td>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 may contain many symbols unnecessary for dynamic linking.</td>
</tr>
</tbody>
</table>

### 11.2.2.2 Additional Section Types

The following additional section types are defined here.

Table 11-2 Additional Section Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHT_GNU_verdef</td>
<td>0x6ffffffd</td>
<td>This section contains the symbol versions that are provided.</td>
</tr>
<tr>
<td>SHT_GNU_verneed</td>
<td>0x6fffffff</td>
<td>This section contains the symbol versions that are required.</td>
</tr>
<tr>
<td>SHT_GNU_versym</td>
<td>0x6fffffff</td>
<td>This section contains the Symbol Version Table.</td>
</tr>
</tbody>
</table>
11.3 Special Sections

11.3.1 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.

11.3.1.1 ELF Special Sections

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.bss</td>
<td>SHT_NOBITS</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.comment</td>
<td>SHT_PROGBITS</td>
<td>0</td>
</tr>
<tr>
<td>.data</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.data1</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.debug</td>
<td>SHT_PROGBITS</td>
<td>0</td>
</tr>
<tr>
<td>.dynamic</td>
<td>SHT_DYNAMIC</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.dynstr</td>
<td>SHT_STRTAB</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.dynsym</td>
<td>SHT_DYNSYM</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.fini</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_EXECINSTR</td>
</tr>
<tr>
<td>.fini_array</td>
<td>SHT_FINI_ARRAY</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.hash</td>
<td>SHT_HASH</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.init</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_EXECINSTR</td>
</tr>
<tr>
<td>.init_array</td>
<td>SHT_INIT_ARRAY</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.interp</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.line</td>
<td>SHT_PROGBITS</td>
<td>0</td>
</tr>
<tr>
<td>.note</td>
<td>SHT_NOTE</td>
<td>0</td>
</tr>
<tr>
<td>.preinit_array</td>
<td>SHT_PREINIT_ARRAY</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.rodata</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_MERGE+SHF_STRINGS</td>
</tr>
<tr>
<td>.rodata1</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_MERGE</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Attributes</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>.shstrtab</td>
<td>SHT_STRTAB</td>
<td>RGE+SHF_STRINGS</td>
</tr>
<tr>
<td>.strtab</td>
<td>SHT_STRTAB</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.symtab</td>
<td>SHT_SYMTAB</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.tbss</td>
<td>SHT_NOBITS</td>
<td>SHF_ALLOC+SHF_WRITE+SHF_TLS</td>
</tr>
<tr>
<td>.tdata</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_WRITE+SHF_TLS</td>
</tr>
<tr>
<td>.text</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_EXECINSTR</td>
</tr>
</tbody>
</table>

.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 of System V ABI Update 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 of System V ABI Update for more information.
.dynsym
This section holds the dynamic linking symbol table, as described in ‘Symbol Table’ of System V ABI Update.

.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 of System V ABI Update 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 of System V ABI Update 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 of System V ABI Update describes.

.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 of System V ABI Update for more information.
.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 of System V ABI Update 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; otherwise, that bit will be off.

.symtab

This section holds a symbol table, as ‘Symbol Table‘ in Chapter 4 of System V ABI Update 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.

.tbss

This section holds uninitialized thread-local data that contribute to the program's memory image. By definition, the system initializes the data with zeros when the data is instantiated for each new execution flow. The section occupies no file space, as indicated by the section type, SHT_NOBITS. Implementations need not support thread-local storage.

.tdata

This section holds initialized thread-local data that contributes to the program's memory image. A copy of its contents is instantiated by the system for each new execution flow. Implementations need not support thread-local storage.

.text

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

### 11.3.1.2 Additional Special Sections

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

<table>
<thead>
<tr>
<th>Table 11-4 Additional Special Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>.ctors</td>
</tr>
<tr>
<td>.data.rel.ro</td>
</tr>
<tr>
<td>.dtors</td>
</tr>
<tr>
<td>.eh_frame</td>
</tr>
</tbody>
</table>
### 11 Object Format

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.eh_frame_hdr</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.gcc_except_table</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.gnu.version</td>
<td>SHT_GNU_versym</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.gnu.version_d</td>
<td>SHT_GNU_verdef</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.gnu.version_r</td>
<td>SHT_GNU_verneed</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.got.plt</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.jcr</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.note.ABI-tag</td>
<td>SHT_NOTE</td>
<td>SHF_ALLOC</td>
</tr>
<tr>
<td>.stab</td>
<td>SHT_PROGBITS</td>
<td>0</td>
</tr>
<tr>
<td>.stabstr</td>
<td>SHT_STRTAB</td>
<td>0</td>
</tr>
</tbody>
</table>

**.ctors**

This section contains a list of global constructor function pointers.

**.data.rel.ro**

This section holds initialized data that contribute to the program's memory image. This section may be made read-only after relocations have been applied.

**.dtors**

This section contains a list of global destructor function pointers.

**.eh_frame**

This section contains information necessary for frame unwinding during exception handling. See Section 11.6.1.

**.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. See Section 11.6.2.

**.gcc_except_table**

This section holds Language Specific Data.

**.gnu.version**

This section contains the Symbol Version Table. See Section 11.7.2.

**.gnu.version_d**

This section contains the Version Definitions. See Section 11.7.3.
.gnu.version_r
This section contains the Version Requirements. See Section 11.7.4.

.got.plt
This section holds the read-only portion of the GLobal Offset Table. This section may be made read-only after relocations have been applied.

.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. See Section 11.8.

.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 information contained in the .stab section.

11.4 Symbol Mapping

11.4.1 Introduction
Symbols in a source program are translated by the compilation system into symbols that exist in the object file.

11.4.1.1 C Language
External C symbols shall be unchanged in an object file's symbol table.

11.5 DWARF Extensions
The LSB does not specify debugging information, however, some additional sections contain information which is encoded using the the encoding as specified by DWARF Debugging Information Format, Revision 2.0.0 with extensions defined here.

Note: The extensions specified here also exist in DWARF Debugging Information Format, Revision 3.0.0 (Draft). It is expected that future versions of the LSB will reference the final version of that document, and that the definitions here will be taken from that document instead of being specified here.

11.5.1 DWARF Exception Header Encoding
The DWARF Exception Header Encoding is used to describe the type of data used in the .eh_frame and .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 11-5 DWARF Exception Header value format

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>© 2007 Linux Foundation</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DW_EH_PE_absptr</td>
<td>0x00</td>
<td>The Value is a literal pointer whose size is determined by the architecture.</td>
</tr>
<tr>
<td>DW_EH_PE_uleb128</td>
<td>0x01</td>
<td>Unsigned value is encoded using the Little Endian Base 128 (LEB128) as defined by DWARF Debugging Information Format, Revision 2.0.0.</td>
</tr>
<tr>
<td>DW_EH_PE_udata2</td>
<td>0x02</td>
<td>A 2 bytes unsigned value.</td>
</tr>
<tr>
<td>DW_EH_PE_udata4</td>
<td>0x03</td>
<td>A 4 bytes unsigned value.</td>
</tr>
<tr>
<td>DW_EH_PE_udata8</td>
<td>0x04</td>
<td>An 8 bytes unsigned value.</td>
</tr>
<tr>
<td>DW_EH_PE_sleb128</td>
<td>0x09</td>
<td>Signed value is encoded using the Little Endian Base 128 (LEB128) as defined by DWARF Debugging Information Format, Revision 2.0.0.</td>
</tr>
<tr>
<td>DW_EH_PE_sdata2</td>
<td>0x0A</td>
<td>A 2 bytes signed value.</td>
</tr>
<tr>
<td>DW_EH_PE_sdata4</td>
<td>0x0B</td>
<td>A 4 bytes signed value.</td>
</tr>
<tr>
<td>DW_EH_PE_sdata8</td>
<td>0x0C</td>
<td>An 8 bytes signed value.</td>
</tr>
</tbody>
</table>

Table 11-6 DWARF Exception Header application

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW_EH_PE_pcrel</td>
<td>0x10</td>
<td>Value is relative to the current program counter.</td>
</tr>
<tr>
<td>DW_EH_PE_textrel</td>
<td>0x20</td>
<td>Value is relative to the beginning of the .text section.</td>
</tr>
<tr>
<td>DW_EH_PE_datarel</td>
<td>0x30</td>
<td>Value is relative to the beginning of the .got or .eh_frame_hdr section.</td>
</tr>
<tr>
<td>DW_EH_PE_funcrel</td>
<td>0x40</td>
<td>Value is relative to the beginning of the function.</td>
</tr>
<tr>
<td>DW_EH_PE_aligned</td>
<td>0x50</td>
<td>Value is aligned to an address unit sized</td>
</tr>
</tbody>
</table>
One special encoding, 0xff (DW_EH_PE_omit), shall be used to indicate that no value is present.

### 11.5.2 DWARF CFI Extensions

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

#### Table 11-7 Additional DWARF Call Frame Instructions

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DW_CFA_expression</td>
<td>0x10</td>
<td>The DW_CFA_expression instruction takes two operands: an unsigned LEB128 value representing a register number, and a DW_FORM_block value representing a DWARF expression. The required action is to establish the DWARF expression as the means by which the address in which the given register contents are found may be computed. The value of the CFA is pushed on the DWARF evaluation stack prior to execution of the DWARF expression. The DW_OP_call2, DW_OP_call4, DW_OP_call_ref and DW_OP_push_object_address DWARF operators (see Section 2.4.1 of DWARF Debugging Information Format, Revision 2.0.0) cannot be used in such a DWARF expression.</td>
</tr>
<tr>
<td>DW_CFA_offset_extended_sf</td>
<td>0x11</td>
<td>The DW_CFA_offset_extended_sf instruction takes two operands: an unsigned LEB128 value</td>
</tr>
<tr>
<td>Name</td>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DW_CFA_def_cfa_sf</strong></td>
<td>0x12</td>
<td>The <code>DW_CFA_def_cfa_sf</code> instruction takes two operands: an unsigned LEB128 value representing a register number and a signed LEB128 factored offset. This instruction is identical to <code>DW_CFA_def_cfa</code> except that the second operand is signed and factored.</td>
</tr>
<tr>
<td><strong>DW_CFA_def_cfa_offset_sf</strong></td>
<td>0x13</td>
<td>The <code>DW_CFA_def_cfa_offset_sf</code> instruction takes a signed LEB128 operand representing a factored offset. This instruction is identical to <code>DW_CFA_def_cfa_offset</code> except that the operand is signed and factored.</td>
</tr>
<tr>
<td><strong>DW_CFA_GNU_args_size</strong></td>
<td>0x2e</td>
<td>The <code>DW_CFA_GNU_args_size</code> instruction takes an unsigned LEB128 operand representing an argument size. This instruction specifies the total of the size of the arguments which have been pushed onto the stack.</td>
</tr>
<tr>
<td><strong>DW_CFA_GNU_negative_offset_extended</strong></td>
<td>0x2f</td>
<td>The <code>DW_CFA_def_cfa_sf</code> instruction takes two operands: an unsigned LEB128 value representing a register number and a signed LEB128 factored offset. This instruction is identical to <code>DW_CFA_def_cfa</code> except that the second operand is signed and factored.</td>
</tr>
</tbody>
</table>
11 Object Format

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>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 instruction is obsoleted by DW_CFA_offset_extended_sf.</td>
</tr>
</tbody>
</table>

11.6 Exception Frames

When using languages that support exceptions, such as C++, additional information must be provided to the runtime environment that describes the call frames that must be unwound during the processing of an exception. This information is contained in the special sections .eh_frame and .eh_framehdr.

**Note:** The format of the .eh_frame section is similar in format and purpose to the .debug_frame section which is specified in DWARF Debugging Information Format, Revision 3.0.0 (Draft). Readers are advised that there are some subtle difference, and care should be taken when comparing the two sections.

11.6.1 The .eh_frame section

The .eh_frame section shall contain 1 or more Call Frame Information (CFI) records. The number of records present shall be determined by size of the section as contained in the section header. Each CFI record contains a Common Information Entry (CIE) record followed by 1 or more Frame Description Entry (FDE) records. Both CIEs and FDEs shall be aligned to an addressing unit sized boundary.

**Table 11-8 Call Frame Information Format**

<table>
<thead>
<tr>
<th>Common Information Entry Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Description Entry Record(s)</td>
</tr>
</tbody>
</table>

11.6.1.1 The Common Information Entry Format

**Table 11-9 Common Information Entry Format**

<table>
<thead>
<tr>
<th>Length</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Length</td>
<td>Optional</td>
</tr>
<tr>
<td>CIE ID</td>
<td>Required</td>
</tr>
<tr>
<td>Version</td>
<td>Required</td>
</tr>
<tr>
<td>Augmentation String</td>
<td>Required</td>
</tr>
</tbody>
</table>
11 Object Format

<table>
<thead>
<tr>
<th>Code Alignment Factor</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Alignment Factor</td>
<td>Required</td>
</tr>
<tr>
<td>Return Address Register</td>
<td>Required</td>
</tr>
<tr>
<td>Augmentation Data Length</td>
<td>Optional</td>
</tr>
<tr>
<td>Augmentation Data</td>
<td>Optional</td>
</tr>
<tr>
<td>Initial Instructions</td>
<td>Required</td>
</tr>
<tr>
<td>Padding</td>
<td></td>
</tr>
</tbody>
</table>

Length
A 4 byte unsigned value indicating the length in bytes of the CIE structure, not including the Length field itself. If Length contains the value 0xffffffff, then the length is contained in the Extended Length field. If Length contains the value 0, then this CIE shall be considered a terminator and processing shall end.

Extended Length
A 8 byte unsigned value indicating the length in bytes of the CIE structure, not including the Length and Extended Length fields.

CIE ID
A 4 byte unsigned value that is used to distinguish CIE records from FDE records. This value shall always be 0, which indicates this record is a CIE.

Version
A 1 byte value that identifies the version number of the frame information structure. This value shall be 1.

Augmentation String
This value is a NUL terminated string that identifies the augmentation to the CIE or to the FDEs associated with this CIE. A zero length string indicates that no augmentation data is present. The augmentation string is case sensitive and shall be interpreted as described below.

Code Alignment Factor
An unsigned LEB128 encoded value that is factored out of all advance location instructions that are associated with this CIE or its FDEs. This value shall be multiplied by the delta argument of an advance location instruction to obtain the new location value.

Data Alignment Factor
A signed LEB128 encoded value that is factored out of all offset instructions that are associated with this CIE or its FDEs. This value shall be multiplied by the register offset argument of an offset instruction to obtain the new offset value.

Augmentation Length
An unsigned LEB128 encoded value indicating the length in bytes of the Augmentation Data. This field is only present if the Augmentation String contains the character ‘z’.
Augmentation Data

A block of data whose contents are defined by the contents of the Augmentation String as described below. This field is only present if the Augmentation String contains the character 'z'. The size of this data is given by the Augmentation Length.

Initial Instructions

Initial set of Call Frame Instructions. The number of instructions is determined by the remaining space in the CIE record.

Padding

Extra bytes to align the CIE structure to an addressing unit size boundary.

11.6.1.1 Augmentation String Format

The Augmentation String indicates the presence of some optional fields, and how those fields should be interpreted. This string is case sensitive. Each character in the augmentation string in the CIE can be interpreted as below:

'z'

A 'z' may be present as the first character of the string. If present, the Augmentation Data field shall be present. The contents of the Augmentation Data shall be interpreted according to other characters in the Augmentation String.

'L'

A 'L' may be present at any position after the first character of the string. This character may only be present if 'z' is the first character of the string. If present, it indicates the presence of one argument in the Augmentation Data of the CIE, and a corresponding argument in the Augmentation Data of the FDE. The argument in the Augmentation Data of the CIE is 1-byte and represents the pointer encoding used for the argument in the Augmentation Data of the FDE, which is the address of a language-specific data area (LSDA). The size of the LSDA pointer is specified by the pointer encoding used.

'P'

A 'P' may be present at any position after the first character of the string. This character may only be present if 'z' is the first character of the string. If present, it indicates the presence of two arguments in the Augmentation Data of the CIE. The first argument is 1-byte and represents the pointer encoding used for the second argument, which is the address of a personality routine handler. The personality routine is used to handle language and vendor-specific tasks. The system unwind library interface accesses the language-specific exception handling semantics via the pointer to the personality routine. The personality routine does not have an ABI-specific name. The size of the personality routine pointer is specified by the pointer encoding used.

'R'

A 'R' may be present at any position after the first character of the string. This character may only be present if 'z' is the first character of the string. If present, The Augmentation Data shall include a 1 byte argument that represents the pointer encoding for the address pointers used in the FDE.
11.6.1.2 The Frame Description Entry Format

Table 11-10 Frame Description Entry Format

<table>
<thead>
<tr>
<th>Length</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Length</td>
<td>Optional</td>
</tr>
<tr>
<td>CIE Pointer</td>
<td>Required</td>
</tr>
<tr>
<td>PC Begin</td>
<td>Required</td>
</tr>
<tr>
<td>PC Range</td>
<td>Required</td>
</tr>
<tr>
<td>Augmentation Data Length</td>
<td>Optional</td>
</tr>
<tr>
<td>Augmentation Data</td>
<td>Optional</td>
</tr>
<tr>
<td>Call Frame Instructions</td>
<td>Required</td>
</tr>
<tr>
<td>Padding</td>
<td></td>
</tr>
</tbody>
</table>

Length

A 4 byte unsigned value indicating the length in bytes of the CIE structure, not including the Length field itself. If Length contains the value 0xffffffff, then the length is contained the Extended Length field. If Length contains the value 0, then this CIE shall be considered a terminator and processing shall end.

Extended Length

A 8 byte unsigned value indicating the length in bytes of the CIE structure, not including the Length field itself.

CIE Pointer

A 4 byte unsigned value that when subtracted from the offset of the the CIE Pointer in the current FDE yields the offset of the start of the associated CIE. This value shall never be 0.

PC Begin

An encoded value that indicates the address of the initial location associated with this FDE. The encoding format is specified in the Augmentation Data.

PC Range

An absolute value that indicates the number of bytes of instructions associated with this FDE.

Augmentation Length

An unsigned LEB128 encoded value indicating the length in bytes of the Augmentation Data. This field is only present if the Augmentation String in the associated CIE contains the character 'z'.

Augmentation Data

A block of data whose contents are defined by the contents of the Augmentation String in the associated CIE as described above. This field is only present if the Augmentation String in the associated CIE contains the character 'z'. The size of this data is given by the Augmentation Length.
Call Frame Instructions

A set of Call Frame Instructions.

Padding

Extra bytes to align the FDE structure to an addressing unit size boundary.

11.6.2 The .eh_frame_hdr section

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 Section 11.5.1.

Table 11-11 .eh_frame_hdr Section Format

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsigned byte</td>
<td>version</td>
</tr>
<tr>
<td>unsigned byte</td>
<td>eh_frame_ptr_enc</td>
</tr>
<tr>
<td>unsigned byte</td>
<td>fde_count_enc</td>
</tr>
<tr>
<td>unsigned byte</td>
<td>table_enc</td>
</tr>
<tr>
<td>encoded</td>
<td>eh_frame_ptr</td>
</tr>
<tr>
<td>encoded</td>
<td>fde_count</td>
</tr>
<tr>
<td></td>
<td>binary search table</td>
</tr>
</tbody>
</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.
11.7 Symbol Versioning

11.7.1 Introduction

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.

11.7.2 Symbol Version Table

The special section .gnu.version which has a section type of SHT_GNU_versym shall contain the Symbol Version Table. This section shall have the same number of entries as the Dynamic Symbol Table in the .dynsym section.

The .gnu.version section shall contain 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 specific 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.

11.7.3 Version Definitions

The special section .gnu.version_d which has a section type of SHT_GNU_verdef shall contain symbol version definitions. The number of entries in this section shall be contained in the DT_VERDEFNUM entry of the Dynamic Section .dynamic. The sh_link member of the section header (see figure 4-8 in the System V ABI) shall point to the section that contains the strings referenced by this section.

The section shall contain an array of Elfxx_Verdef structures, as described in Figure 11-1, optionally followed by an array of Elfxx_Vernaux structures, as defined in Figure 11-2.

typedef struct {
    Elfxx_Half    vd_version;
    Elfxx_Half    vd_flags;
    Elfxx_Half    vd_ndx;
    Elfxx_Half    vd_cnt;
    Elfxx_Word    vd_hash;
} Elfxx_Verdef;
Elfxx_Word vd_aux;
Elfxx_Word vd_next;
} Elfxx_Verdef;

Figure 11-1 Version Definition Entries

vd_version
Version revision. This field shall be set to 1.

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 in bytes to a corresponding entry in an array of Elfxx_Verdaux structures as defined in Figure 11-2

vd_next
Offset to the next verdef entry, in bytes.

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

Figure 11-2 Version Definition Auxiliary Entries

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.

11.7.4 Version Requirements

The special section .gnu.version_r which has a section type of SHT_GNU_verneed shall contain required symbol version definitions. The number of entries in this section shall be contained in the DT_VERNEEDNUM entry of the Dynamic Section .dynamic. The sh_link member of the section header (see figure 4-8 in System V ABI) shall point to the section that contains the strings referenced by this section.

The section shall contain an array of Elfxx_Verneed structures, as described in Figure 11-3, optionally followed by an array of Elfxx_Vernaux structures, as defined in Figure 11-4.

typedef struct {
    Elfxx_Half vn_version;
    Elfxx_Half vn_cnt;
} Elfxx_Verneed;
11 Object Format

Figure 11-3 Version Needed Entries

```c
typedef struct {
    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.

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

Figure 11-4 Version Needed Auxiliary Entries

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.

11.7.5 Startup Sequence

When loading a sharable object the system shall analyze version definition data
from the loaded object to assure that it meets the version requirements of the
calling object. This step is referred to as definition testing. The dynamic loader
shall retrieve the entries in the caller’s Elfxx_Verneed array and attempt to find matching definition information in the loaded Elfxx_Verdef table.

Each object and dependency shall be tested in turn. If a symbol definition is missing and the vna_flags bit for VER_FLG_WEAK is not set, the loader shall return an error and exit. If the vna_flags bit for VER_FLG_WEAK is set in the Elfxx_Vernaux entry, and the loader shall issue a warning and continue operation.

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 shall be made available.

11.7.6 Symbol Resolution

When symbol versioning is used in an object, relocations extend 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.

If the high order bit (bit number 15) of the version symbol is set, the object cannot be used and the static linker shall ignore the symbol's presence in the object.

When an object with a reference and an object with the definition are being linked, the following rules shall govern the result:

• The object with the reference and the object with the definitions both use versioning. All described matching is processed in this case. A fatal error shall be 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 does not use versioning, while the object with the definitions does. In this instance, only the definitions 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 cases where the static linker was not used, such as in calls to dlopen(), a version that does not have the base definition index shall be acceptable if it is the only version for which the symbol is defined.

• The object with the reference uses versioning, but the object with the definitions specifies none. A matching symbol shall be accepted in this case. A fatal error shall be triggered if a corruption in the required symbols list obscures an outdated object file and causes a match on the object filename in the Elfxx_Verneed entry.

• Neither the object with the reference nor the object with the definitions use versioning. The behavior in this instance shall default to pre-existing symbol rules.

11.8 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.
12 Dynamic Linking

12.1 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 further required by this specification and the relevant architecture specific part of ISO/IEC 23360.

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

12.2 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 12-1 Linux Segment Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT_GNU_EH_FRAME</td>
<td>0x6474e550</td>
</tr>
<tr>
<td>PT_GNU_STACK</td>
<td>0x6474e551</td>
</tr>
<tr>
<td>PT_GNU_RELRO</td>
<td>0x6474e552</td>
</tr>
</tbody>
</table>

PT_GNU_EH_FRAME

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

PT_GNU_STACK

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

PT_GNU_RELRO

The array element specifies the location and size of a segment which may be made read-only after relocations have been processed.

12.3 Dynamic Entries

12.3.1 Introduction

As described in System V ABI, if an object file participates in dynamic linking, its program header table shall have an element of type PT_DYNAMIC. This `segment' contains the .dynamic section. A special symbol, _DYNAMIC, labels the section, which contains an array of the following structures.

```c
typedef struct {
    Elf32_Sword     d_tag;
    union {
        Elf32_Word      d_val;
        Elf32.Addr      d_ptr;
    } d_un;
} Elf32_Dyn;

extern Elf32_Dyn        _DYNAMIC[];
```
typedef struct {
    Elf64_Sxword    d_tag;
    union {
        Elf64_Xword     d_val;
        Elf64_Addr      d_ptr;
    } d_un;
} Elf64_Dyn;

extern Elf64_Dyn    _DYNAMIC[];

Figure 12-1 Dynamic Structure

For each object with this type, d_tag controls the interpretation of d_un.

12.3.2 Dynamic Entries

12.3.2.1 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 relocations

DT_REL
   Address of Rel relocations

DT_RELAC
   Address of Rela relocations

DT_RELACNT
   Size of one Rela relocation

DT_RELACS
   Total size of Rela relocations

DT_RELENT
   Size of one Rel relocation

DT_RELSZ
   Total size of Rel relocations

DT_RPATH
   Library search path

DT_SONAME
   Name of shared object

DT_STRSZ
   Size of string table

DT_STRTAB
   Address of string table

DT_SYMENT
   Size of one symbol table entry

DT_SYMTAB
   Address of symbol table

DT_TEXTREL
   Relocation might modify .text

### 12.3.2.2 Additional Dynamic Entries

An LSB conforming object may also use the following additional Dynamic Entry types.
12 Dynamic Linking

ISO/IEC 23360 Part 1:2007(E)

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_RUNPATH
null-terminated library search path string

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

DT_VALRNGLO
Entries which fall between DT_VALRNGHI & DT_VALRNGLO use the

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.
III Base Libraries
13 Base Libraries

13.1 Introduction

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

- libc
- libm
- libgcc_s
- libdl
- librt
- libcrypt
- libpam

There are three main parts to the definition of each of these libraries.

The "Interfaces" section defines the required library name and version, and the required public symbols (interfaces and global data), as well as symbol versions, if any.

The "Interface Definitions" section provides complete or partial definitions of certain interfaces where either this specification is the source specification, or where there are variations from the source specification. If an interface definition requires one or more header files, one of those headers shall include the function prototype for the interface.

For source definitions of interfaces which include a reference to a header file, the contents of such header files form a part of the specification. The "Data Definitions" section provides the binary-level details for the header files from the source specifications, such as values for macros and enumerated types, as well as structure layouts, sizes and padding, etc. These data definitions, although presented in the form of header files for convenience, should not be taken a representing complete header files, as they are a supplement to the source specifications. Application developers should follow the guidelines of the source specifications when determining which header files need to be included to completely resolve all references.

**Note:** While the Data Definitions supplement the source specifications, this specification itself does not require conforming implementations to supply any header files.

13.2 Program Interpreter

The Program Interpreter is specified in the appropriate architecture specific part of ISO/IEC 23360.

13.3 Interfaces for libc

Table 13-1 defines the library name and shared object name for the libc library

<table>
<thead>
<tr>
<th>Library:</th>
<th>libc</th>
</tr>
</thead>
</table>

© 2007 Linux Foundation
The behavior of the interfaces in this library is specified by the following specifications:

[LFS] Large File Support
[LSB] This Specification
[SUSv2] SUSv2
[SVID.3] SVID Issue 3
[SVID.4] SVID Issue 4

13.3.1 RPC

13.3.1.1 Interfaces for RPC

An LSB conforming implementation shall provide the generic functions for RPC specified in Table 13-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-2 libc - RPC Function Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>svcudp_create [LSB]</td>
<td>xdr_accepted_reply [SVID.3]</td>
<td>xdr_array [SVID.3]</td>
<td>xdr_bool [SVID.3]</td>
</tr>
<tr>
<td>xdr_int [SVID.3]</td>
<td>xdr_long [SVID.3]</td>
<td>xdr_opaque [SVID.3]</td>
<td>xdr_opaque_aut h [SVID.3]</td>
</tr>
</tbody>
</table>
13 Base Libraries

<table>
<thead>
<tr>
<th>xdr_void</th>
<th>xdr_wrapstring</th>
<th>xdrmem_create</th>
<th>xdrrec_create</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SVID.3]</td>
<td>[SVID.3]</td>
<td>[SVID.3]</td>
<td>[SVID.3]</td>
</tr>
</tbody>
</table>

**13.3.2 System Calls**

**13.3.2.1 Interfaces for System Calls**

An LSB conforming implementation shall provide the generic functions for System Calls specified in Table 13-3, with the full mandatory functionality as described in the referenced underlying specification.

<table>
<thead>
<tr>
<th>Table 13-3 libc - System Calls Function Interfaces</th>
</tr>
</thead>
</table>

© 2007 Linux Foundation
### 13.3.3 Standard I/O

#### 13.3.3.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the generic functions for Standard I/O specified in Table 13-5, with the full mandatory functionality as described in the referenced underlying specification.

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IO_feof</td>
<td>[LSB]</td>
<td>_IO_getc</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_IO_putc</td>
<td>[LSB]</td>
<td>_IO_puts</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

---

An LSB conforming implementation shall provide the generic deprecated functions for System Calls specified in Table 13-4, with the full mandatory functionality as described in the referenced underlying specification.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fstatfs</td>
<td>[LSB]</td>
<td>getdtablesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>statfs</td>
<td>[LSB]</td>
<td>getpagesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>statvfs</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>time</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>utime</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>utimes</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>wait</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>wait4</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>waitid</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

---

#### Table 13-4 libc - System Calls Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fstatfs</td>
<td>[LSB]</td>
<td>getdtablesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>statfs</td>
<td>[LSB]</td>
<td>getpagesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>time</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>utime</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>utimes</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>wait</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>wait4</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
<tr>
<td>waitid</td>
<td>[LSB]</td>
<td>settimeofday</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

---

#### Table 13-5 libc - Standard I/O Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IO_feof</td>
<td>[LSB]</td>
<td>_IO_getc</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_IO_putc</td>
<td>[LSB]</td>
<td>_IO_puts</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

---
An LSB conforming implementation shall provide the generic data interfaces for Standard I/O specified in Table 13-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-6 libc - Standard I/O Data Interfaces

|----------------|----------------|----------------|

13.3.4 Signal Handling

13.3.4.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the generic functions for Signal Handling specified in Table 13-7, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-7 libc - Signal Handling Function Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bsd_signal [SUSv3]</td>
<td>psignal [LSB]</td>
<td>raise [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigaction</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigaddset</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigalert</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigaltstack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigdelset</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigemptyset</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigfillset</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sighold</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigignore</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>siginterrupt</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigisemptyset</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigismember</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>siglongjmp</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigorset</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigpause</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigpending</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigprocmask</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigqueue</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigrelse</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigreturn</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigset</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigsuspend</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigtimedwait</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigwait</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sigwaitinfo</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic deprecated functions for Signal Handling specified in Table 13-8, with the full mandatory functionality as described in the referenced underlying specification.

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

**Table 13-8 libc - Signal Handling Deprecated Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigpause</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Signal Handling specified in Table 13-9, with the full mandatory functionality as described in the referenced underlying specification.

**Table 13-9 libc - Signal Handling Data Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sys_siglist</td>
</tr>
</tbody>
</table>

### 13.3.5 Localization Functions

#### 13.3.5.1 Interfaces for Localization Functions

An LSB conforming implementation shall provide the generic functions for Localization Functions specified in Table 13-10, with the full mandatory functionality as described in the referenced underlying specification.

**Table 13-10 libc - Localization Functions Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind_textdomain</td>
<td></td>
</tr>
<tr>
<td>_codeset</td>
<td>[LSB]</td>
</tr>
<tr>
<td>bindtextdomain</td>
<td>[LSB]</td>
</tr>
<tr>
<td>catclose</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>catgets</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>catopen</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>dcgettext</td>
<td>[LSB]</td>
</tr>
<tr>
<td>dcngettext</td>
<td>[LSB]</td>
</tr>
<tr>
<td>dgettext</td>
<td>[LSB]</td>
</tr>
<tr>
<td>duplocale</td>
<td>[GLIBC _2.3]</td>
</tr>
<tr>
<td>freelocale</td>
<td>[GLIBC _2.3]</td>
</tr>
<tr>
<td>gettext</td>
<td>[LSB]</td>
</tr>
<tr>
<td>iconv</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iconv_close</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iconv_open</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>localeconv</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>
newlocale(GLIBC_2.3) [LSB]  ngettext [LSB]  nl_langinfo [SUSv3]  setlocale [SUSv3]

textdomain [LSB]  uselocale(GLIBC_2.3) [LSB]

An LSB conforming implementation shall provide the generic data interfaces for Localization Functions specified in Table 13-11, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-11 libc - Localization Functions Data Interfaces

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>_nl_msg_cat_cntr</td>
</tr>
</tbody>
</table>

13.3.6 Posix Spawn Option

13.3.6.1 Interfaces for Posix Spawn Option

An LSB conforming implementation shall provide the generic functions for Posix Spawn Option specified in Table 13-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-12 libc - Posix Spawn Option Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>posix_spawn [SUSv3]</td>
</tr>
<tr>
<td>posixspawn_file</td>
</tr>
<tr>
<td>_actions_addclose</td>
</tr>
<tr>
<td>[SUSv3]</td>
</tr>
<tr>
<td>posixspawn_file</td>
</tr>
<tr>
<td>_actions_adddup</td>
</tr>
<tr>
<td>2 [SUSv3]</td>
</tr>
<tr>
<td>posixspawn_file</td>
</tr>
<tr>
<td>_actions_destroy</td>
</tr>
<tr>
<td>[SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_getpgroup [SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_getschedparam</td>
</tr>
<tr>
<td>[SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_init [SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_setflags [SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_setgroup [SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_setschedparam</td>
</tr>
<tr>
<td>[SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_setsigmask [SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_setsigdefault</td>
</tr>
<tr>
<td>[SUSv3]</td>
</tr>
<tr>
<td>posixspawnattr</td>
</tr>
<tr>
<td>_setsigmask [SUSv3]</td>
</tr>
<tr>
<td>posix_spawnp [SUSv3]</td>
</tr>
</tbody>
</table>

13.3.7 Posix Advisory Option

13.3.7.1 Interfaces for Posix Advisory Option

An LSB conforming implementation shall provide the generic functions for Posix Advisory Option specified in Table 13-13, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-13 libc - Posix Advisory Option Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>posix_fadvise</td>
</tr>
<tr>
<td>posix_fallocate</td>
</tr>
<tr>
<td>posix_madvise</td>
</tr>
<tr>
<td>posix_memalign</td>
</tr>
</tbody>
</table>
13.3.8 Socket Interface

13.3.8.1 Interfaces for Socket Interface

An LSB conforming implementation shall provide the generic functions for Socket Interface specified in Table 13-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-14 libc - Socket Interface Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>__h_errno_location</td>
<td></td>
<td></td>
<td>accept</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>connect</td>
<td>SUSv3</td>
<td></td>
<td>bind</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>gethostid</td>
<td>SUSv3</td>
<td></td>
<td>gethostname</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>getsockopt</td>
<td></td>
<td></td>
<td>getpeername</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>getsockname</td>
<td>SUSv3</td>
<td></td>
<td>if_freenameindex</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>getsockopt</td>
<td></td>
<td></td>
<td>if_indextoname</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>if_nameindex</td>
<td>SUSv3</td>
<td></td>
<td>listen</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>recvfrom</td>
<td>SUSv3</td>
<td></td>
<td>recv</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>send</td>
<td>SUSv3</td>
<td></td>
<td>sendmsg</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>setsockopt</td>
<td></td>
<td></td>
<td>shutdown</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>socket</td>
<td>SUSv3</td>
<td></td>
<td>sockatmark</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>socketpair</td>
<td>SUSv3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Socket Interface specified in Table 13-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-15 libc - Socket Interface Data Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>in6addr_any</td>
<td></td>
<td>SUSv3</td>
<td>in6addr_loopback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.3.9 Wide Characters

13.3.9.1 Interfaces for Wide Characters

An LSB conforming implementation shall provide the generic functions for Wide Characters specified in Table 13-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-16 libc - Wide Characters Function Interfaces

| Function                  | LSB | | Function                  | LSB | |
|---------------------------|-----| |---------------------------|-----| |
| __wcstod_internal         |     | | __wcstof_internal         |     | |
| __wcstol_internal         |     | | __wcstold.Internal         |     | |
| __wcstoul_internal        |     | | btowc                     | SUSv3 | |
|                           |     | | fgetwc                    | SUSv3 | |
|                           |     | | fgetws                    | SUSv3 | |
| fputwc                    | SUSv3 | | fputws                    | SUSv3 | |
|                           |     | | fwide                     | SUSv3 | |
|                           |     | | fwprintf                  | SUSv3 | |
### 13.3.10 String Functions

#### 13.3.10.1 Interfaces for String Functions

An LSB conforming implementation shall provide the generic functions for String Functions specified in Table 13-17, with the full mandatory functionality as described in the referenced underlying specification.
An LSB conforming implementation shall provide the generic deprecated functions for String Functions specified in Table 13-18, with the full mandatory functionality as described in the referenced underlying specification.

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

### Table 13-18 libc - String Functions Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__mempcpy [LSB]</td>
<td>__rawmemchr [LSB]</td>
</tr>
<tr>
<td>__strtod_internal [LSB]</td>
<td>__strtof_internal [LSB]</td>
</tr>
<tr>
<td>__strtold_internal [LSB]</td>
<td>__strtol_internal [LSB]</td>
</tr>
<tr>
<td>__xpg_strerror_r( GLIBC_2.3.4) [LSB]</td>
<td>bcmp [SUSv3]</td>
</tr>
<tr>
<td>ffs [SUSv3]</td>
<td>index [SUSv3]</td>
</tr>
<tr>
<td>memcmp [SUSv3]</td>
<td>memcpyp [SUSv3]</td>
</tr>
<tr>
<td>memset [SUSv3]</td>
<td>rindex [SUSv3]</td>
</tr>
<tr>
<td>strcasecmp [SUSv3]</td>
<td>strcoll [SUSv3]</td>
</tr>
<tr>
<td>strcmp [SUSv3]</td>
<td>strlen [SUSv3]</td>
</tr>
<tr>
<td>strdup [SUSv3]</td>
<td>strftime [SUSv3]</td>
</tr>
<tr>
<td>strpbrk [SUSv3]</td>
<td>strxfrm [SUSv3]</td>
</tr>
<tr>
<td>strftime [SUSv3]</td>
<td>swab [SUSv3]</td>
</tr>
<tr>
<td>strerror_r [LSB]</td>
<td>strptime [LSB]</td>
</tr>
</tbody>
</table>

© 2007 Linux Foundation
13.3.11 IPC Functions

13.3.11.1 Interfaces for IPC Functions

An LSB conforming implementation shall provide the generic functions for IPC Functions specified in Table 13-19, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-19 libc - IPC Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftok</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>msgctl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>msgget</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>msgsnd</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>semctl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>semget</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>semop</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>shmat</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>shmdt</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>shmat</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>shmdt</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

13.3.12 Regular Expressions

13.3.12.1 Interfaces for Regular Expressions

An LSB conforming implementation shall provide the generic functions for Regular Expressions specified in Table 13-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-20 libc - Regular Expressions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>regcomp</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>regerror</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>regexec</td>
<td>[LSB]</td>
</tr>
<tr>
<td>regfree</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

13.3.13 Character Type Functions

13.3.13.1 Interfaces for Character Type Functions

An LSB conforming implementation shall provide the generic functions for Character Type Functions specified in Table 13-21, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-21 libc - Character Type Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ctype_b_loc(GLIBC_2.3)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_ctype_get_mb_cur_max</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_ctype_tolower_loc(GLIBC_2.3)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_ctype_toupper_loc(GLIBC_2.3)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_tolower</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>_toupper</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isalnum</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isalpha</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isascii</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iscntrl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isdigit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isgraph</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>islower</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isprint</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ispunct</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isspace</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>isupper</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswalnum</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswalpha</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswblank</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswcntrl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswctype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswdigit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswgraph</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswlower</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswprint</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswpunct</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswspace</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswupper</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iswxdigit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>issdigit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>toascii</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>
13.3.14 Time Manipulation

13.3.14.1 Interfaces for Time Manipulation

An LSB conforming implementation shall provide the generic functions for Time Manipulation specified in Table 13-22, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-22 libc - Time Manipulation Function Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ualarm [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Time Manipulation specified in Table 13-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-23 libc - Time Manipulation Data Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>timezone [SUSv3]</td>
<td>tzname [SUSv3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.3.15 Terminal Interface Functions

13.3.15.1 Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the generic functions for Terminal Interface Functions specified in Table 13-24, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-24 libc - Terminal Interface Functions Function Interfaces

|---------------------|---------------------|-----------------|---------------------|
13.3.16 System Database Interface

13.3.16.1 Interfaces for System Database Interface

An LSB conforming implementation shall provide the generic functions for System Database Interface specified in Table 13-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-25 libc - System Database Interface Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>endgrent</td>
<td></td>
<td>[SUSv3]</td>
<td>endprotoent</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>endpwent</td>
<td></td>
<td>[SUSv3]</td>
<td>endservent</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>endutent</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
<td>getgrent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getggid_r</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>getgnam</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getgposium</td>
<td></td>
<td>[SUSv3]</td>
<td>getgrgiulist</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyaddr</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>gethostbyaddr_r</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyname</td>
<td></td>
<td>[SUSv3]</td>
<td>gethostbyname2</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyname2_r</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
<td>getprotobyname</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getpwent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>getpwnam</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getpwnam_r</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>getservbyname</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getpwuid</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>getservbyport</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getservent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>getutent</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getutent_r</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
<td>getutxent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getutxline</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>pututxline</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setgrent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>setprotoent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setpwent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>setservent</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setutent</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
<td>utmpname</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic deprecated functions for System Database Interface specified in Table 13-26, with the full mandatory functionality as described in the referenced underlying specification.

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

Table 13-26 libc - System Database Interface Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>gethostbyaddr</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyaddr_r</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyname</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyname2</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gethostbyname2_r</td>
<td>[LSB]</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

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

13.3.17.1 Interfaces for Language Support

An LSB conforming implementation shall provide the generic functions for Language Support specified in Table 13-27, with the full mandatory functionality as described in the referenced underlying specification.

<table>
<thead>
<tr>
<th>__libc_start_main [LSB]</th>
<th>__register_atfork (GLIBC_2.3.2) [LSB]</th>
</tr>
</thead>
</table>

Table 13-27 libc - Language Support Function Interfaces

13.3.18 Large File Support

13.3.18.1 Interfaces for Large File Support

An LSB conforming implementation shall provide the generic functions for Large File Support specified in Table 13-28, with the full mandatory functionality as described in the referenced underlying specification.

<table>
<thead>
<tr>
<th>__fxstat64 [LSB]</th>
<th>__lxstat64 [LSB]</th>
<th>__xstat64 [LSB]</th>
<th>creat64 [LFS]</th>
</tr>
</thead>
<tbody>
<tr>
<td>fgetpos64 [LFS]</td>
<td>fopen64 [LFS]</td>
<td>freopen64 [LFS]</td>
<td>fseek64 [LFS]</td>
</tr>
<tr>
<td>fsetpos64 [LFS]</td>
<td>fstatfs64 [LSB]</td>
<td>fstatvfs64 [LFS]</td>
<td>ftello64 [LFS]</td>
</tr>
<tr>
<td>ftruncate64 [LFS]</td>
<td>ftw64 [LFS]</td>
<td>getrlimit64 [LFS]</td>
<td>lockf64 [LFS]</td>
</tr>
<tr>
<td>mktemp64 [LFS]</td>
<td>mmap64 [LFS]</td>
<td>nftw64 [LFS]</td>
<td>posix_fadvise64 [LSB]</td>
</tr>
<tr>
<td>nftw64 [LFS]</td>
<td>statfs64 [LSB]</td>
<td>statvfs64 [LFS]</td>
<td>tmpfile64 [LFS]</td>
</tr>
<tr>
<td>posix_fallocate64 [LFS]</td>
<td>readdir64 [LFS]</td>
<td>readdir64_r [LSB]</td>
<td>truncate64 [LFS]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic deprecated functions for Large File Support specified in Table 13-29, with the full mandatory functionality as described in the referenced underlying specification.

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

<table>
<thead>
<tr>
<th>__fxstat64 [LSB]</th>
<th>__lxstat64 [LSB]</th>
<th>__xstat64 [LSB]</th>
<th>creat64 [LFS]</th>
</tr>
</thead>
<tbody>
<tr>
<td>fstatfs64 [LSB]</td>
<td>statfs64 [LSB]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 13-29 libc - Large File Support Deprecated Function Interfaces

13.3.19 Standard Library

13.3.19.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the generic functions for Standard Library specified in Table 13-30, with the full mandatory functionality as described in the referenced underlying specification.
<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT</td>
</tr>
<tr>
<td>_assert_fail</td>
</tr>
<tr>
<td>_cxa_atexit</td>
</tr>
<tr>
<td>_cxa_finalize</td>
</tr>
<tr>
<td>_errno_location</td>
</tr>
<tr>
<td>_fpending</td>
</tr>
<tr>
<td>_getpagesize</td>
</tr>
<tr>
<td>_isinf</td>
</tr>
<tr>
<td>_isinfl</td>
</tr>
<tr>
<td>_isnanl</td>
</tr>
<tr>
<td>_isnaff</td>
</tr>
<tr>
<td>_isnalf</td>
</tr>
<tr>
<td>_longjmp</td>
</tr>
<tr>
<td>_setjmp</td>
</tr>
<tr>
<td>a4l</td>
</tr>
<tr>
<td>abort</td>
</tr>
<tr>
<td>abs</td>
</tr>
<tr>
<td>atof</td>
</tr>
<tr>
<td>atoi</td>
</tr>
<tr>
<td>atol</td>
</tr>
<tr>
<td>atoll</td>
</tr>
<tr>
<td>basename</td>
</tr>
<tr>
<td>bsearch</td>
</tr>
<tr>
<td>calloc</td>
</tr>
<tr>
<td>closelog</td>
</tr>
<tr>
<td>confstr</td>
</tr>
<tr>
<td>cuserid</td>
</tr>
<tr>
<td>daemon</td>
</tr>
<tr>
<td>dirname</td>
</tr>
<tr>
<td>div</td>
</tr>
<tr>
<td>drand48</td>
</tr>
<tr>
<td>erand48</td>
</tr>
<tr>
<td>err</td>
</tr>
<tr>
<td>error</td>
</tr>
<tr>
<td>errx</td>
</tr>
<tr>
<td>fcvt</td>
</tr>
<tr>
<td>fmtmsg</td>
</tr>
<tr>
<td>fnmatch</td>
</tr>
<tr>
<td>fpathconf</td>
</tr>
<tr>
<td>free</td>
</tr>
<tr>
<td>freeaddrinfo</td>
</tr>
<tr>
<td>ftrylockfile</td>
</tr>
<tr>
<td>ftw</td>
</tr>
<tr>
<td>funlockfile</td>
</tr>
<tr>
<td>gai_strerror</td>
</tr>
<tr>
<td>gcvt</td>
</tr>
<tr>
<td>getaddrinfo</td>
</tr>
<tr>
<td>getcver</td>
</tr>
<tr>
<td>getdomainname</td>
</tr>
<tr>
<td>getenv</td>
</tr>
<tr>
<td>getopt</td>
</tr>
<tr>
<td>getopt_long</td>
</tr>
<tr>
<td>getopt_long_only</td>
</tr>
<tr>
<td>getsubopt</td>
</tr>
<tr>
<td>gettimeofday</td>
</tr>
<tr>
<td>grantpt</td>
</tr>
<tr>
<td>hcreate</td>
</tr>
<tr>
<td>hdestroy</td>
</tr>
<tr>
<td>hsearch</td>
</tr>
<tr>
<td>htonl</td>
</tr>
<tr>
<td>htons</td>
</tr>
<tr>
<td>imaxabs</td>
</tr>
<tr>
<td>imaxdiv</td>
</tr>
<tr>
<td>inet_addr</td>
</tr>
<tr>
<td>inet_pton</td>
</tr>
<tr>
<td>inet_aton</td>
</tr>
<tr>
<td>inet_ntoa</td>
</tr>
<tr>
<td>inet_ntop</td>
</tr>
<tr>
<td>initstate</td>
</tr>
<tr>
<td>insque</td>
</tr>
<tr>
<td>isatty</td>
</tr>
<tr>
<td>isblank</td>
</tr>
<tr>
<td>jrand48</td>
</tr>
<tr>
<td>ldiv</td>
</tr>
<tr>
<td>lfind</td>
</tr>
<tr>
<td>labs</td>
</tr>
<tr>
<td>llabs</td>
</tr>
<tr>
<td>lldiv</td>
</tr>
<tr>
<td>longjmp</td>
</tr>
<tr>
<td>lrand48</td>
</tr>
<tr>
<td>lsearch</td>
</tr>
<tr>
<td>makecontext</td>
</tr>
<tr>
<td>malloc</td>
</tr>
<tr>
<td>memmem</td>
</tr>
<tr>
<td>mkstemp</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for Standard Library specified in Table 13-31, with the full mandatory functionality as described in the referenced underlying specification.

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

**Table 13-31 libc - Standard Library Deprecated Function Interfaces**

<table>
<thead>
<tr>
<th>basename [LSB]</th>
<th>getdomainname [LSB]</th>
<th>inet_aton [LSB]</th>
</tr>
</thead>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Standard Library specified in Table 13-32, with the full mandatory functionality as described in the referenced underlying specification.

**Table 13-32 libc - Standard Library Data Interfaces**

<table>
<thead>
<tr>
<th>__environ [LSB]</th>
<th>__environ [LSB]</th>
<th>__sys_errlist [LSB]</th>
<th>environ [SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>optopt [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 13.4 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

13.4.1 arpa/inet.h

extern uint32_t htonl(uint32_t);
extern uint16_t htons(uint16_t);
extern in_addr_t inet_addr(const char *);
extern int inet_aton(const char *, struct in_addr *);
extern char *inet_ntoa(struct in_addr);
extern const char *inet_ntop(int, const void *, char *,
socklen_t);
extern int inet_ntop(int, const char *, void *);
extern uint32_t ntohl(uint32_t);
extern uint16_t ntohs(uint16_t);

13.4.2 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.

extern void __assert_fail(const char *, const char *, unsigned
int,
const char *);

13.4.3 ctype.h

extern int _tolower(int);
extern int _toupper(int);
extern int isalnum(int);
extern int isalpha(int);
extern int isascii(int);
extern int iscntrl(int);
extern int isdigit(int);
extern int isgraph(int);
extern int islower(int);
extern int isprint(int);
extern int ispunct(int);
extern int isspace(int);
extern int isupper(int);
extern int isxdigit(int);
extern int toascii(int);
extern int tolower(int);
extern int toupper(int);
extern int isblank(int);
extern const short **__ctype_b_loc(void);
extern const int32_t **__ctype_toupper_loc(void);
extern const int32_t **__ctype_tolower_loc(void);

13.4.4 dirent.h

typedef struct __dirstream DIR;

struct dirent {
    long int 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];
};
extern int readdir64_r(DIR *, struct dirent64 *, struct dirent64 *);
extern void rewinddir(DIR *);
extern void seekdir(DIR *, long int);
extern long int telldir(DIR *);
extern int closedir(DIR *);
extern DIR *opendir(const char *);
extern struct dirent *readdir(DIR *);
extern struct dirent64 *readdir64(DIR *);
extern int readdir_r(DIR *, struct dirent *, struct dirent **);

13.4.5 err.h

extern void err(int, const char *, ...);
extern void errx(int, const char *, ...);
extern void warn(const char *, ...);
extern void warnx(const char *, ...);
extern void error(int, int, const char *, ...);

13.4.6 errno.h

ISO POSIX (2003) requires that each error value shall be unique, with permission for EAGAIN and EWOULDBLOCK possibly having the same value. This specification also requires that ENOTSUP and EOPNOTSUPP have the same value.

Note: A defect report against ISO POSIX (2003) has been filed to request that specification also permit these two symbols to have the same value.

#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 ENOBUFFS 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 EHOSTUNREACH 118
#define ENAVAIL 119
#define ENOMEM 12
#define EISNAM 120
#define EREMOTEIO 121
#define EDQUOT 122
#define EENOSYS 123
#define EEMEDIUMTYPE 124
#define ECANCELLED 125
#define EFAULT 126
#define EACCES 127
#define ENOTBLK 128
#define EBUSY 129
#define EEXIST 130
#define EXDEV 131
#define ENODEV 132
#define ENOTDIR 133
#define EISDIR 134
#define EINVAL 135
#define ENFILE 136
#define EMFILE 137
#define ENOTTY 138
#define ETXTBSY 139
#define EFBIG 140
#define ENOSPC 141
#define ESPIPE 142
#define ESRCH 143
#define EROFS 144
#define EMLINK 145
#define EPIPE 146
#define EDOM 147
#define ERANGE 148
#define EDEADLK 149
#define ENAMETOOLONG 150
#define ENOLCK 151
#define ENOSYS 152
#define ENOTEMPTY 153
#define EINTR 154
#define ELOOP 155
#define ENOMSG 156
#define EIDRM 157
#define ECHRNG 158
#define EL2NSYNC 159
#define EL3HLT 160
#define EL3RST 161
#define ELNRNG 162
#define EUNATCH 163
#define EIO 164
#define ENOANO 165
#define EBADRQC 166
#define EBADSLT 167
#define EBFONT 168
#define EL3HLT 46
#define EL3RST 47
#define ELNRNG 48
#define EUNATCH 49
#define EIO 5
#define ENOANO 55
#define EBADRQC 56
#define EBADSLT 57
#define EBFONT 59
#define ENXIO   6
#define ENOSTR  60
#define ENODATA 61
#define ETIME   62
#define ENOSR   63
#define ENONET  64
#define ENOPKG  65
#define EREMOTE 66
#define ENOLINK 67
#define EADV    68
#define ESRMNT  69
#define EZBIG   7
#define ECOMM   70
#define EPROTO  71
#define EMULTIHOP  72
#define EDOTDOT 73
#define EOVERFLOW 75
#define ENOTUNIQ 76
#define EBADFD  77
#define EREMCHG 78
#define ELIBACC 79
#define ENOEXEC 8
#define ELIBBAD 80
#define ELIBSCN 81
#define ELIBMAX 82
#define ELIBEXEC 83
#define EILSEQ  84
#define ERESTART 85
#define ESTRPIPE 86
#define EUSERS  87
#define ENOTSOCK 88
#define EDESTADDRREQ 89
#define EBADF  9
#define EMSGSIZE 90
#define EPROTOTYPE 91
#define EOPNOTSUPP 92
#define EPFNOSUPPORT 93
#define ESOCKTNOSUPPORT 94
#define EPROTONOSUPPORT 95
#define EAFNOSUPPORT 96
#define EADDRINUSE 97
#define EADDRNOTAVAIL 98
#define EWOULDBLOCK EAGAIN
#define ENOTSUP EOPNOTSUPP
extern int *__errno_location(void);

13.4.7 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_VAL O_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;
    off64_t l_start;
    off64_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_SETSIG 10
#define F_GETSIG 11
#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

extern int posix_fadvise(int, off_t, off_t, int);
extern int posix_fallocate(int, off_t, off_t);
extern int posix_fadvise64(int, off64_t, off64_t, int);
extern int posix_fallocate64(int, off64_t, off64_t);
extern int creat(const char *, mode_t);
extern int creat64(const char *, mode_t);
extern int fcntl(int, int, ...);
extern int open(const char *, int, ...);
extern int open64(const char *, int, ...);

13.4.8 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)

extern int fmtmsg(long int, const char *, int, const char *,
                   const char *,
                   const char *);
13.4.9 fnmatch.h

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

extern int fnmatch(const char *, const char *, int);

13.4.10 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 = 1,
    FTW_MOUNT = 2,
    FTW_CHDIR = 4,
    FTW_DEPTH = 8
};

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);

extern int ftw(const char *, __ftw_func_t, int);
extern int ftw64(const char *, __ftw64_func_t, int);
extern int nftw(const char *, __nftw_func_t, int, int);
extern int nftw64(const char *, __nftw64_func_t, int, int);

13.4.11 getopt.h

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

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

extern int getopt_long(int, char *const, const char *,
                        const struct option *, int *);
extern int getopt_long_only(int, char *const, const char *,
                             const struct option *, int *);

13.4.12 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_NOSPACE    (1<<9)
#define GLOB_ABORTED    (1<<10)
#define GLOB_NOMATCH    (1<<11)
#define GLOB_NOSYS      (1<<12)
#define GLOB_ALTDIRFUNC (1<<14)

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_readdir) (void *);
    void *(*gl_opendir) (const char *);
    int (*gl_lstat) (const char *, struct stat *);
    int (*gl_stat) (const char *, struct stat *);
} glob64_t;

extern int glob(const char *, int, int (*gl_stat) (const char *
p1, int p2)
    , glob_t *);
extern int glob64(const char *, int,
                   int (*gl_stat) (const char *p1, int p2)
                   , glob64_t *);
13 Base Libraries

ISO/IEC 23360 Part 1:2007(E)

extern void globfree(glob_t *);
extern void globfree64(glob64_t *);

13.4.13 grp.h

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

extern void endgrent(void);
extern struct group *getgrent(void);
extern struct group *getgrgid(gid_t);
extern struct group *getgrnam(const char *);
extern int initgroups(const char *, gid_t);
extern void setgrent(void);
extern int setgroups(size_t, const gid_t *);
extern int getgrgid_r(gid_t, struct group *, char *, size_t,
    struct group **);
extern int getgrnam_r(const char *, struct group *, char *,
    size_t, struct group **);
extern int getgrouplist(const char *, gid_t, gid_t *, int *);

13.4.14 iconv.h

typedef void *iconv_t;
extern size_t iconv(iconv_t, char **, size_t *, char **, size_t *);
extern int iconv_close(iconv_t);
extern iconv_t iconv_open(const char *, const char *);

13.4.15 inttypes.h

typedef lldiv_t imaxdiv_t;

extern intmax_t strtoimax(const char *, char **, int);
extern uintmax_t strtoumax(const char *, char **, int);
extern intmax_t wcstoimax(const wchar_t *, wchar_t **, int);
extern uintmax_t wcstoumax(const wchar_t *, wchar_t **, int);
extern intmax_t imaxabs(intmax_t);
extern imaxdiv_t imaxdiv(intmax_t, intmax_t);

13.4.16 langinfo.h

#define ABDAY_1 0x20000
#define ABDAY_2 0x20001
#define ABDAY_3 0x20002
#define ABDAY_4 0x20003
#define ABDAY_5 0x20004
#define ABDAY_6 0x20005
#define ABDAY_7 0x20006
#define DAY_1 0x20007
#define DAY_2 0x20008
#define DAY_3 0x20009
#define DAY_4 0x2000A
#define DAY_5 0x2000B
#define DAY_6 0x2000C
#define DAY_7 0x2000D
#define ABMON_1 0x2000E
#define ABMON_2 0x2000F
#define ABMON_3 0x20010
#define ABMON_4 0x20011
#define ABMON_5 0x20012
#define ABMON_6 0x20013
#define ABMON_7 0x20014
#define ABMON_8 0x20015
#define ABMON_9 0x20016
#define ABMON_10 0x20017
#define ABMON_11 0x20018
#define ABMON_12 0x20019

#define MON_1 0x2001A
#define MON_2 0x2001B
#define MON_3 0x2001C
#define MON_4 0x2001D
#define MON_5 0x2001E
#define MON_6 0x2001F
#define MON_7 0x20020
#define MON_8 0x20021
#define MON_9 0x20022
#define MON_10 0x20023
#define MON_11 0x20024
#define MON_12 0x20025

#define AM_STR 0x20026
#define PM_STR 0x20027

#define D_T_FMT 0x20028
#define D_FMT 0x20029
#define T_FMT 0x2002A
#define T_FMT_AMPM 0x2002B

#define ERA 0x2002C
#define ERA_D_FMT 0x2002E
#define ALT_DIGITS 0x2002F
#define ERA_D_T_FMT 0x20030
#define ERA_T_FMT 0x20031

#define CODESET 14

#define CRNCYSTR 0x4000F
#define RADIXCHAR 0x10000
#define THOUSEP 0x10001
#define YESEXPR 0x50000
#define NOEXPR 0x50001
#define YESSTR 0x50002
#define NOSTR 0x50003

extern char *nl_langinfo(nl_item);

13.4.17 libgen.h

#define basename __xpg_basename

extern char *dirname(char *);
extern char *__xpg_basename(char *);

13.4.18 libintl.h
extern char *bindtextdomain(const char *, const char *);
extern char *dcgettext(const char *, const char *, int);
extern char *dgettext(const char *, const char *);
extern char *gettext(const char *);
extern char *textdomain(const char *);
extern char *bind_textdomain_codeset(const char *, const char *);
extern char *dcngettext(const char *, const char *, const char *,
                      unsigned long int, int);
extern char *dngettext(const char *, const char *, const char *,
                      unsigned long int);
extern char *ngettext(const char *, const char *, unsigned long
                      int);

13.4.19 limits.h

#define LLONG_MIN       (~LLONG_MAX-1LL)
#define _POSIX_AIO_MAX  1
#define _POSIX_QLIMIT   1
#define _POSIX2_BC_STRING_MAX   1000
#define _POSIX2_CHARCLASS_NAME_MAX  14
#define _POSIX_NAME_MAX 14
#define _POSIX_UIO_MAXIOV 16
#define ULLONG_MAX      18446744073709551615ULL
#define _POSIX2_COLL_WEIGHTS_MAX        2
#define _POSIX_AIO_LISTIO_MAX   2
#define _POSIX_OPEN_MAX 20
#define _POSIX_CLOCKRES_MIN     20000000
#define CHARCLASS_NAME_MAX      2048
#define LINE_MAX        2048
#define _POSIX2_BC_DIM_MAX      2048
#define _POSIX2_LINE_MAX        2048
#define _POSIX_CHILD_MAX        25
#define COLL_WEIGHTS_MAX        255
#define _POSIX2_RE_DUP_MAX      255
#define _POSIX_MAX_CANON        255
#define _POSIX_MAX_INPUT        255
#define _POSIX_RE_DUP_MAX       255
#define _POSIX_SYMLOOP_MAX      255
#define OPEN_MAX        256
#define _POSIX_PATH_MAX 256
#define _POSIX2_EXPR_NEST_MAX   32
#define _POSIX_DELAYTIMER_MAX   32
#define _POSIX_MQ_PRIO_MAX      32
#define _POSIX_TIMER_MAX        32
#define _POSIX_SEM_VALUE_MAX    32767
#define _POSIX_SSIZE_MAX        32767
#define PATH_MAX        4096
#define _POSIX_ARG_MAX  4096
#define _POSIX_PIPE_BUF 512
#define _POSIX2_T2NAME_MAX     6
#define _POSIX2_LINK_MAX      8
#define _POSIX2_MQ_OPEN_MAX    8
#define _POSIX2_NGROUPS_MAX    8
#define _POSIX2_RTSIG_MAX      8
#define _POSIX2_STREAM_MAX     8
#define _POSIX2_SYMLOOP_MAX    8
#define _POSIX2_LOGIN_NAME_MAX 9
#define _POSIX2_TTY_NAME_MAX   9
#define LLONG_MAX      ~9223372036854775807ULL
#define _POSIX2_BC_BASE_MAX   99
#define _POSIX2_BC_SCALE_MAX  99
#define SSIZE_MAX       LONG_MAX
13.4.20 locale.h

struct lconv {
    char *decimal_point;
    char *thousands_sep;
    char *grouping;
    char *int_curr_symbol;
    char *currency_symbol;
    char *mon_decimal_point;
    char *mon_thousands_sep;
    char *mon_grouping;
    char *positive_sign;
    char *negative_sign;
    char *int_frac_digits;
    char *frac_digits;
    char *p_cs_precedes;
    char *p_sep_by_space;
    char *n_cs_precedes;
    char *n_sep_by_space;
    char *p_sign_posn;
    char *n_sign_posn;
    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;
};

#define LC_GLOBAL_LOCALE ((locale_t) -1L)
#define LC_CTYPE 0
#define LC_NUMERIC 1
#define LC_TIME 2
#define LC_IDENTIFICATION 12
13.4.21 monetary.h

extern ssize_t strfmon(char *, size_t, const char *, ...);

13.4.22 net/if.h

#define IF_NAMESIZE 16
#define IFF_UP 0x01
#define IFF_BROADCAST 0x02
#define IFF_DEBUG 0x04
#define IFF_LOOPBACK 0x08
#define IFF_POINTOPOINT 0x10
#define IFF_PROMISC 0x100
#define IFF_MULTICAST 0x1000
#define IFF_NOTRAILERS 0x20
#define IFF_RUNNING 0x40
#define IFF_NOARP 0x80

struct if_nameindex {
    unsigned int if_index;
    char *if_name;
};

struct ifaddr {
    struct sockaddr ifa_addr;
    union {
        struct sockaddr ifu_broadaddr;
        struct sockaddr ifu_dstaddr;
    } ifa_ifu;
    void *ifa_ifp;
    void *ifa_next;
};

#define IFNAMSIZ IF_NAMESIZE

struct ifreq {
    union {
        char ifrn_name[IFNAMSIZ];
    } ifr_ifrn;
    union {
        struct sockaddr ifru_addr;
        struct sockaddr ifru_broadaddr;
        struct sockaddr ifru_dstaddr;
        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;
};

extern void if_freenameindex(struct if_nameindex *);
extern char *if_indextoname(unsigned int, char *);
extern struct if_nameindex *if_nameindex(void);
extern unsigned int if_nametoindex(const char *);

13.4.23 netdb.h

#define NETDB_INTERNAL -1
#define NETDB_SUCCESS 0
#define HOST_NOT_FOUND 1
#define IPPROTO_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;
};

struct hostent {
  char *h_name;
  char **h_aliases;
  int h_addrtype;
  int h_length;
  char **h_addr_list;
};

struct protoent {
  char *p_name;
  char **p_aliases;
  int p_proto;
};

struct netent {
  char *n_name;
  char **n_aliases;
  int n_addrtype;
  unsigned int n_net;
};

#define AI_PASSIVE 0x0001
#define AI_CANONNAME 0x0002
#define AI_NUMERICHOST 0x0004

struct addrinfo {
  int ai_flags;
  int ai_family;
  int ai_socktype;
  int ai_protocol;
  socklen_t ai_addrlen;
  struct sockaddr *ai_addr;
  char *ai_canonname;
  struct addrinfo *ai_next;
};

#define NI_NUMERICHOST 1
#define NI_DGRAM 16
#define NI_NUMERICSERV 2
#define NI_NOFQDN 4
#define NI_NAMEREQD 8

#define EAI_BADFLAGS -1
#define EAI_MEMORY -10
#define EAI_SYSTEM -11
#define EAI_NONAME -2
#define EAI_AGAIN -3
#define EAI_FAIL -4
#define EAI_NODATA -5
#define EAI_FAMILY -6
#define EAI_SOCKTYPE -7
#define EAI_SERVICE -8
#define EAI_ADDRFAMILY -9

extern int gethostbyname2_r(const char *, int, struct hostent *,
  char *,
  size_t, struct hostent **, int *);
extern void endprotoent(void);
extern void endservent(void);
extern void freeaddrinfo(struct addrinfo *);
extern int getaddrinfo(const char *, const char *, const struct
  addrinfo *,
  struct addrinfo **);
extern struct hostent *gethostbyaddr(const void *, socklen_t, int);
extern struct hostent *gethostbyname(const char *);
extern struct hostent *gethostbyname2(const char *, int);
extern struct protoent *getprotobyname(const char *);
extern struct protoent *getprotobynumber(int);
extern struct protoent *getprotoent(void);
extern struct servent *getservbyname(const char *, const char *);
extern struct servent *getservbyport(int, const char *);
extern struct servent *getservent(void);
extern void setprotoent(int);
extern void setservent(int);
extern int *__h_errno_location(void);
extern int gethostbyaddr_r(const void *, socklen_t, int, struct hostent *, char *, struct hostent **, int);
extern int gethostbyname_r(const char *, struct hostent *, char *, struct hostent **, int *);

13.4.24 netinet/in.h

#define IPPROTO_IP      0
#define IPPROTO_ICMP    1
#define IPPROTO_UDP     17
#define IPPROTO_ICMPV6  58
#define IPPROTO_RAW     255
#define IPPROTO_IPV6    41
#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 IP_TTL 2
#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
#define IP_OPTIONS 4

struct ipv6_mreq {
    struct in6_addr ipv6mr_multiaddr;
    int ipv6mr_interface;
};

struct ip_mreq {
    struct in_addr imr_multiaddr;
    struct in_addr imr_interface;
};

extern int bindresvport(int, struct sockaddr_in *);
extern const struct in6_addr in6addr_any;
extern const struct in6_addr in6addr_loopback;

13.4.25 netinet/ip.h

#define IPTOS_LOWCAST 0x02
#define IPTOS_RELIABILITY 0x04
#define IPTOS_THROUGHPUT 0x08
#define IPTOS_LOWDELAY 0x10
#define IPTOS_TOS_MASK 0x1e
#define IPTOS_MINCOST IPTOS_LOWCAST

#define IPTOS_PREC_MASK 0xe0

13.4.26 netinet/tcp.h

#define TCP_NODELAY 1
#define SOL_TCP 6

13.4.27 netinet/udp.h

#define SOL_UDP 17

13.4.28 nl_types.h
#define NL_CAT_LOCALE   1
#define NL_SETD 1

typedef void *nl_catd;

typedef int nl_item;
extern int catclose(nl_catd);
extern char *catgets(nl_catd, int, int, const char *);
extern nl_catd catopen(const char *, int);

13.4.29 poll.h

extern int poll(struct pollfd *, nfds_t, int);

13.4.30 pty.h

extern int openpty(int *, int *, char *, struct termios *,
                     struct winsize *);
extern int forkpty(int *, char *, struct termios *, struct
                     winsize *);

13.4.31 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;
};
extern void endpwent(void);
extern struct passwd *getpwent(void);
extern struct passwd *getpwnam(const char *);
extern struct passwd *getpwuid(uid_t);
extern void setpwent(void);
extern int getpwnam_r(const char *, struct passwd *, char *,
                        size_t, struct passwd **);
extern int getpwuid_r(uid_t, struct passwd *, char *, size_t, struct
                        passwd **);

13.4.32 regex.h

#define RE_DUP_MAX     (0x7fff)

typedef unsigned long int reg_syntax_t;

typedef struct re_pattern_buffer {
    unsigned char *buffer;
    unsigned long int allocated;
    unsigned long int used;
    reg_syntax_t syntax;
    char *fastmap;
    char *translate;
    size_t re_nsub;
    unsigned int can_be_null:1;
    unsigned int regs_allocated:2;
    unsigned int fastmap_accurate:1;
    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_ICASE       (REG_EXTENDED<<1)
#define REG_NEWLINE     (REG_ICASE<<1)
#define REG_NOSUB       (REG_NEWLINE<<1)
#define REG_EXTENDED    1
#define REG_NOTEOL      (1<<1)
#define REG_NOTBOL      1

typedef enum {
    REG_ENOSYS = -1,
    REG_NOERROR = 0,
    REG_NOMATCH = 1,
    REG_BADPAT = 2,
    REG_ECOLLATE = 3,
    REG_ECTYPE = 4,
    REG_EESCAPE = 5,
    REG_ESUBREG = 6,
    REG_EBRACK = 7,
    REG_EPARAREN = 8,
    REG_EBRACE = 9,
    REG_BADBR = 10,
    REG_ENGRANGE = 11,
    REG_ESPACE = 12,
    REG_EBADRPT = 13,
    REG_EEND = 14,
    REG_ESIZE = 15,
    REG_EPARAREN = 16
} reg_errcode_t;
extern int regcomp(regex_t *, const char *, int);
extern size_t regerror(int, const regex_t *, char *, size_t);
extern int regexec(const regex_t *, const char *, size_t, regmatch_t, int);
extern void regfree(regex_t *);

13.4.33 rpc/auth.h

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

enum auth_stat {
    AUTH_OK = 0,
    AUTH_BADCRED = 1,
    AUTH_REJECTEDCRED = 2,
    AUTH_BADVERF = 3,
    AUTH_REJECTEDVERF = 4,
    AUTH_TOOWEAK = 5,
    AUTH_INVALIDRESP = 6,
    AUTH_FAILED = 7
};
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_marshall) (struct AUTH *, XDR *);
    int (*ah_validate) (struct AUTH *, struct opaque_auth *);
    int (*ah_refresh) (struct AUTH *);
    void (*ah_destroy) (struct AUTH *);
};

extern struct AUTH *authnone_create(void);
extern int key_decryptsession(char *, union des_block *);
extern bool_t xdr_opaque_auth(XDR *, struct opaque_auth *);

13.4.34 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_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
#define clnt_call(rh, proc, xargs, argsp, xres, resp, secs)     \
    (*((rh)->cl_ops->cl_call)(rh, proc, xargs, argsp, xres, resp, secs))

enum clint_stat {
    RPC_SUCCESS = 0,
    RPC_CANTENCODERARGS = 1,
    RPC_CANTDECODERES = 2,
    RPC_CANTSEND = 3,
    RPC_CANTRECV = 4,
    RPC_TIMEDOUT = 5,
};
RPC_VERSMISMATCH = 6,
RPC_AUTHERROR = 7,
RPC_PROGUNAVAIL = 8,
RPC_PROGVERSMMATCH = 9,
RPC_PROGUNAVAIL = 10,
RPC_CANTDECODEARGS = 11,
RPC_SYSTEMERROR = 12,
RPC_NOBROADCAST = 21,
RPC_UNKNOWNHOST = 13,
RPC_UNKNOWNPROTO = 17,
RPC_UNKNOWNADDR = 19,
RPC_RPCBFAILURE = 14,
RPC_PROGNOTREGISTERED = 15,
RPC_N2AXLATEFAILURE = 22,
RPC_FAILED = 16,
RPC_INTR = 18,
RPC_TLIERROR = 20,
RPC_UDERROR = 23,
RPC_INPROGRESS = 24,
RPC_STALERACHANDLE = 25
};

struct rpc_err {
    enum clnt_stat re_status;
    union {
        int RE_errno;
        enum auth_stat RE_why;
        struct {
            u_long low;
            u_long high;
        } RE_vers;
        struct {
            long int s1;
            long int s2;
        } RE_lb;
    } ru;
};

typedef struct CLIENT {
    struct AUTH *cl_auth;
    struct clnt_ops *cl_ops;
    caddr_t cl_private;
} CLIENT;

struct clnt_ops {
    enum clnt_stat (*cl_call) (struct CLIENT *, u_long, xdrproc_t, caddr_t, xdrproc_t, caddr_t, struct timeval);
    void (*cl_abort) (void);
    void (*cl_geterr) (struct CLIENT *, struct rpc_err *);
    bool_t(*cl_freeres) (struct CLIENT *, xdrproc_t, caddr_t);
    void (*cl_destroy) (struct CLIENT *);
    bool_t(*cl_control) (struct CLIENT *, int, char *);
};

extern struct CLIENT *clnt_create(const char *, const u_long, const char *, const u_long, xdrproc_t, caddr_t, xdrproc_t, caddr_t, struct timeval);
void (*cl_abort) (void);
void (*cl_geterr) (struct CLIENT *, struct rpc_err *);
bool_t(*cl_freeres) (struct CLIENT *, xdrproc_t, caddr_t);
void (*cl_destroy) (struct CLIENT *);
bool_t(*cl_control) (struct CLIENT *, int, char *);
extern struct CLIENT *clnt_create(const char *, const u_long, const char *, const u_long, xdrproc_t, caddr_t, xdrproc_t, caddr_t, struct timeval);
exten void clnt_pcreateerror(const char *);
exten void clnt_perrno(enum clnt_stat);
exten void clnt_perror(const char *, struct CLIENT *);
exten char *clnt_pcreateerror(const char *);
exten char *clnt_spcreateerror(const char *);
exten char *clnt_perrno(enum clnt_stat);
exten char *clnt_sperrno(const char *);
exten char *clnt_sperror(const char *, struct CLIENT *);
13.4.35 rpc/pmap_clnt.h

extern u_short pmap_getport(struct sockaddr_in *, const u_long, const u_long, u_int);
extern bool_t pmap_set(const u_long, const u_long, int, u_short);
extern bool_t pmap_unset(u_long, u_long);

13.4.36 rpc/rpc_msg.h

enum msg_type {
    CALL = 0,
    REPLY = 1
};
enum reply_stat {
    MSG_ACCEPTED = 0,
    MSG_DENIED = 1
};
enum accept_stat {
    SUCCESS = 0,
    PROG_UNAVAIL = 1,
    PROG_MISMATCH = 2,
    PROC_UNAVAIL = 3,
    GARBAGE_ARGS = 4,
    SYSTEM_ERR = 5
};
enum reject_stat {
    RPC_MISMATCH = 0,
    AUTH_ERROR = 1
};
struct accepted_reply {
    struct opaque_auth ar_verf;
    enum accept_stat ar_stat;
    union {
        struct {
            unsigned long int low;
            unsigned long int high;
        } AR_versions;
        struct {
            caddr_t where;
            xdrproc_t proc;
        } AR_results;
    } ru;
};
struct rejected_reply {
    enum reject_stat rj_stat;
    union {
        struct {
            unsigned long int low;
            unsigned long int 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 int cb_rpcvers;
    unsigned long int cb_prog;
    unsigned long int cb_vers;
    unsigned long int cb_proc;
    struct opaque_auth cb_cred;
    struct opaque_auth cb_verf;
};

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

extern bool_t xdr_accepted_reply(XDR *, struct accepted_reply *);
extern bool_t xdr_callhdr(XDR *, struct rpc_msg *);
extern bool_t xdr_callmsg(XDR *, struct rpc_msg *);
extern bool_t xdr_rejected_reply(XDR *, struct rejected_reply *);
extern bool_t xdr_replymsg(XDR *, struct rpc_msg *);

13.4.37 rpc/svc.h

#define svc_getcaller(x)        (&(x)->xp_raddr)
#define svc_destroy(xprt)       (*(xprt)->xp_ops->xp_destroy)(xprt)
#define svc_recv(xprt,msg)      (*(xprt)->xp_ops->xp_recv)((xprt), (msg))
#define svc_reply(xprt,msg)     (*(xprt)->xp_ops->xp_reply)((xprt), (msg))
#define svc_stat(xprt)  (*(xprt)->xp_ops->xp_stat)(xprt)
#define RPC_ANYSOCK     -1
#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))

enum xprt_stat {
    XPRT_DIED,
    XPRT_MOREREQS,
    XPRT_IDLE
};

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);
};

extern void svc_getreqset(fd_set *);
extern bool_t svc_register(SVCXPRT *, rpcprog_t, rpcvers_t,
                           __dispatch_fn_t, rpcprot_t);
extern void svc_run(void);
extern bool_t svc_sendreply(SVCXPRT *, xdrproc_t, caddr_t);
extern void svcerr_auth(SVCXPRT *, enum auth_stat);
extern void svcerr_decode(SVCXPRT *);
extern void svcerr_noproc(SVCXPRT *);
extern void svcerr_noprog(SVCXPRT *);
extern void svcerr_progvers(SVCXPRT *, rpcvers_t, rpcvers_t);
extern void svcerr_systemerr(SVCXPRT *);
extern void svcerr_weakauth(SVCXPRT *);
extern SVCXPRT *svctcp_create(int, u_int, u_int);
extern SVCXPRT *ssvcudp_create(int);

### 13.4.38 rpc/types.h

typedef int bool_t;
typedef int enum_t;
typedef unsigned long int rpcprog_t;
typedef unsigned long int rpcvers_t;
typedef unsigned long int rpcproc_t;
typedef unsigned long int rpcprot_t;

### 13.4.39 rpc/xdr.h

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;

typedef struct xdr_ops {
    bool_t(*x_getlong) (XDR * __xdrs, long int *__lp);
    bool_t(*x_putlong) (XDR * __xdrs, long int *__lp);
    bool_t(*x_getbytes) (XDR * __xdrs, caddr_t __addr, u_int __len);
    bool_t(*x_putchar) (XDR * __xdrs, char *__addr, u_int __len);
    u_int(*x_getposn) (XDR * __xdrs);
    bool_t(*x_setposn) (XDR * __xdrs, u_int __pos);
} XDR_ops;
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;
};

extern void xdrstdio_create(XDR *, FILE *, enum xdr_op);
extern bool_t xdr_array(XDR *, caddr_t *, u_int *, u_int, u_int,
xdrproc_t);
extern bool_t xdr_bool(XDR *, bool_t *);
extern bool_t xdr_bytes(XDR *, char **, u_int *, u_int);
extern bool_t xdr_char(XDR *, char *);
extern bool_t xdr_double(XDR *, double *);
extern bool_t xdr_enum(XDR *, enum_t *);
extern bool_t xdr_float(XDR *, float *);
extern void xdr_free(xdrproc_t, char *);
extern bool_t xdr_int(XDR *, int *);
extern bool_t xdr_long(XDR *, long int *);
extern bool_t xdr_opaque(XDR *, caddr_t, u_int);
extern bool_t xdr_pointer(XDR *, char **, u_int, xdrproc_t);
extern bool_t xdr_reference(XDR *, char *, u_int, xdrproc_t);
extern bool_t xdr_short(XDR *, short *);
extern bool_t xdr_string(XDR *, char *, u_int);
extern bool_t xdr_u_char(XDR *, u_char *);
extern bool_t xdr_u_int(XDR *, u_int *);
extern bool_t xdr_u_long(XDR *, u_long *);
extern bool_t xdr_u_short(XDR *, u_short *);
extern bool_t xdr_union(XDR *, enum_t *, char *,
    const struct xdr_discrim *, xdrproc_t);
extern bool_t xdr_vector(XDR *, char *, u_int, u_int, xdrproc_t);
extern bool_t xdr_void(void);
extern bool_t xdr_wrapstring(XDR *, char **);
extern void xdrmem_create(XDR *, caddr_t, u_int, enum xdr_op);
extern void xdrrec_create(XDR *, u_int, u_int, caddr_t,
    int (*proc) (char *p1, char *p2, int p3),
    int (*proc) (char *p1, char *p2, int p3);
}
extern bool_t xdrrec_eof(XDR *);

13.4.40 sched.h

#define SCHED_OTHER     0
#define SCHED_FIFO      1
#define SCHED_RR        2

struct sched_param {
    int sched_priority;
};

extern int sched_get_priority_max(int);
extern int sched_get_priority_min(int);
extern int sched_getparam(pid_t, struct sched_param *);
extern int sched_setscheduler(pid_t);
extern int sched_rr_get_interval(pid_t, struct timespec *);
extern int sched_setparam(pid_t, const struct sched_param *);
extern int sched_setscheduler(pid_t, int, const struct
sched_param *);
extern int sched_yield(void);
13.4.41 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);
extern int hcreate(size_t);
extern ENTRY *hsearch(ENTRY, ACTION);
extern void insque(void *, void *);
extern void *lfind(const void *, const void *, size_t *, size_t, __compar_fn_t);
extern void *lsearch(const void *, void *, size_t *, size_t, __compar_fn_t);
extern void remque(void *);
extern void hdestroy(void);
extern void *tdelete(const void *, void **, __compar_fn_t);
extern void *tfind(const void *, void *const *, __compar_fn_t);
extern void *tsearch(const void *, void **, __compar_fn_t);
extern void twalk(const void *, __action_fn_t);

13.4.42 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;
extern int __sigsetjmp(jmp_buf, int);
extern void longjmp(jmp_buf, int);
extern void siglongjmp(sigjmp_buf, int);
extern void _longjmp(jmp_buf, int);
extern int _setjmp(jmp_buf);

13.4.43 signal.h

#define sigpause __xpg_sigpause
#define _SIGSET_NWORDS  (1024/(8*sizeof(unsigned long)))
#define SIGRTMAX        (__libc_current_sigrtmax ()
#define SIGRTMIN        (__libc_current_sigrtmin ()
#define SIG_BLOCK       0
#define SIG_UNBLOCK     1
#define SIG_SETMASK     2
#define NSIG  65

typedef int sig_atomic_t;

typedef void (*sighandler_t) (int);

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

#define SIGHUP   1
#define SIGUSR1   10
#define SIGSEGV   11
#define SIGUSR2   12
#define SIGPIPE   13
#define SIGALRM   14
#define SIGTERM   15
#define SIGSTKFLT  16
#define SIGCHLD   17
#define SIGCONT   18
#define SIGSTOP   19
#define SIGINT    2
#define SIGTSTP   20
#define SIGTIN    21
#define SIGTTOUT  22
#define SIGURG    23
#define SIGXCPU   24
#define SIGXFSZ   25
#define SIGVTALRM 26
#define SIGPROF   27
#define SIGWINCH  28
#define SIGIO     29
#define SIGQUIT   3
#define SIGPWR    30
#define SIGSYS    31
#define SIGUNUSED 31
#define SIGILL    4
#define SIGTRAP   5
#define SIGIOT    6
#define SIGBUS    7
#define SIGFPE    8
#define SIGKILL   9
#define SIGCLD    SIGCHLD
#define SIGPOLL   SIGIO

#define SV_ONSTACK  (1<<0)
#define SV_INTERRUPT (1<<1)
#define SV_RESETHAND (1<<2)

typedef union sigval {
    int sival_int;
    void *sival_ptr;
} sigval_t;

#define SIGEV_SIGNAL  0
#define SIGEV_NONE    1
#define SIGEV_THREAD  2
#define SIGEV_MAX_SIZE 64

typedef struct sigevent {
    sigval_t sigev_value;
    int sigev_signo;
    int sigev_notify;
    union {

int _pad[SIGEV_PAD_SIZE];
struct {
    void (*_function) (sigval_t);
    void * _attribute;
} _sigev_thread;
} _sigev_un;
} sigevent_t;
#define SI_MAX_SIZE 128
#define si_pid _sifields._kill._pid
#define si_uid _sifields._kill._uid
#define si_value _sifields._rt._sigval
#define si_int _sifields._rt._sigval.sival_int
#define si_ptr _sifields._rt._sigval.sival_ptr
#define si_status _sifields._sigchld._status
#define si_stime _sifields._sigchld._stime
#define si_utime _sifields._sigchld._utime
#define si_addr _sifields._sigfault._addr
#define si_band _sifields._sigpoll._band
#define si_fd _sifields._sigpoll._fd
#define si_timer1 _sifields._timer._timer1
#define si_timer2 _sifields._timer._timer2

typedef struct siginfo {
    int si_signo;
    int si_errno;
    int si_code;
    union {
        int _pad[SI_PAD_SIZE];
        struct {
            pid_t _pid;
            uid_t _uid;
        } _kill;
        struct {
            unsigned int _timer1;
            unsigned int _timer2;
        } _timer;
        struct {
            pid_t _pid;
            uid_t _uid;
            sigval_t _sigval;
        } _rt;
        struct {
            pid_t _pid;
            uid_t _uid;
            int _status;
            clock_t _utime;
            clock_t _stime;
        } _sigchld;
        struct {
            void * _addr;
        } _sigfault;
        struct {
            int _band;
            int _fd;
        } _sigpoll;
    } _sifields;
} siginfo_t;
#define SI_QUEUE -1
#define SI_TIMER -2
#define SI_MESGQ -3
#define SI_ASYNCIO -4
#define SI_SIGIO -5
#define SI_TKILL -6
#define SI_ASYNCNL -60
#define SI_USER 0
#define SI_KERNEL 0x80
#define ILL_ILLOPC 1
#define ILL_ILLOPN 2
#define ILL_ILLADR 3
#define ILL_ILLTRP 4
#define ILL_PRVOPC 5
#define ILL_PRVREG 6
#define ILL_COPROC 7
#define ILL_BADSTK 8
#define FPE_INTDIV 1
#define FPE_INTOVF 2
#define FPE_FLTDIV 3
#define FPE_FLTOVF 4
#define FPE_FLTUND 5
#define FPE_FLTRES 6
#define FPE_FLTINV 7
#define FPE_FLTSUB 8
#define SEGV_MAPERR 1
#define SEGV_ACCERR 2
#define BUS_ADRALN 1
#define BUS_ADRERR 2
#define BUS_OBJERR 3
#define TRAP_BRKPT 1
#define TRAP_TRACE 2
#define CLD_EXITED 1
#define CLD_KILLED 2
#define CLD_DUMPED 3
#define CLD_TRAPPED 4
#define CLD_STOPPED 5
#define CLD_CONTINUED 6
#define POLL_IN 1
#define POLL_OUT 2
#define POLL_MSG 3
#define POLL_ERR 4
#define POLL_PRI 5
#define POLL_HUP 6

typedef struct {
    unsigned long int sig[_SIGSET_NWORDS];
} sigset_t;

#define SA_NOCLDSTOP 0x00000001
#define SA_NOCLDWAIT 0x00000002
#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

extern int __libc_current_sigrtmax(void);
extern int __libc_current_sigrtmin(void);
extern sighandler_t __sysv_signal(int, sighandler_t);
extern int killpg(pid_t, int);
extern void psignal(int, const char *);
extern int raise(int);
extern int sigaddset(sigset_t *, int);
extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
extern int sigdelset(sigset_t *, int);
extern int sigemptyset(sigset_t *);
extern int sigfillset(sigset_t *);
extern int sighold(int);
extern int sigignore(int);
extern int siginterrupt(int, int);
extern int sigisemptyset(const sigset_t *);
extern int sigismember(const sigset_t *, int);
extern int sigormask(sigset_t *, const sigset_t *, const sigset_t *);
extern int sigpending(sigset_t *);
extern int sigprocmask(int, const sigset_t *, sigset_t *);
extern int sigreturn(struct sigcontext *);
extern int sigsuspend(const sigset_t *);
extern int sigpause(int);

13.4.44 spawn.h

#define POSIX_SPAWN_RESETIDS 0x01
#define POSIX_SPAWN_SETPGROUP 0x02
#define POSIX_SPAWN_SETSIGDEF 0x04
#define POSIX_SPAWN_SETSIGMASK 0x08
#define POSIX_SPAWN_SETSCHEDPARAM 0x10
#define POSIX_SPAWN_SETSCHEDULER 0x20

typedef struct {
  int __allocated;
  int __used;
  struct __spawn_action *__actions;
  int __pad[16];
} posix_spawn_file_actions_t;

typedef struct {
  short __flags;
  pid_t __pgrp;
  sigset_t __sd;
  sigset_t __ss;
} posix_spawn_attr_t;
struct sched_param __sp;
int __policy;
int __pad[16];
} posix_spawnattr_t;

extern int posix_spawn(pid_t *, const char *,
const posix_spawn_file_actions_t *,
const posix_spawnattr_t *, char *const,
char *const);

extern int posix_spawn_file_actions_addclose(posix_spawn_file_actions_t *,
int);
extern int posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t *,
int, int);
extern int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t *,
int, const char *,
int,
mode_t);
extern int posix_spawn_file_actions_destroy(posix_spawn_file_actions_t *);
extern int posix_spawn_file_actions_init(posix_spawn_file_actions_t *);
extern int posix_spawnattr_destroy(posix_spawnattr_t *);
extern int posix_spawnattr_getflags(const posix_spawnattr_t *,
short int *);
extern int posix_spawnattr_getpgroup(const posix_spawnattr_t *,
pid_t *);
extern int posix_spawnattr_getschedparam(const posix_spawnattr_t *,
struct sched_param *);
extern int posix_spawnattr_getschedpolicy(const posix_spawnattr_t *,
int *);
extern int posix_spawnattr_getsigdefault(const posix_spawnattr_t *,
sigset_t *);
extern int posix_spawnattr_getsigmask(const posix_spawnattr_t *,
sigset_t *);
extern int posix_spawnattr_init(posix_spawnattr_t *);
extern int posix_spawnattr_setflags(posix_spawnattr_t *, short int);
extern int posix_spawnattr_setpgroup(posix_spawnattr_t *, pid_t);
extern int posix_spawnattr_setschedparam(posix_spawnattr_t *,
const struct sched_param *);
extern int posix_spawnattr_setschedpolicy(posix_spawnattr_t *,
int);
extern int posix_spawnattr_setsigdefault(posix_spawnattr_t *,
const sigset_t *);
extern int posix_spawnattr_setsigmask(posix_spawnattr_t *,
const sigset_t *);
extern int posix_spawn(pid_t *, const char *,
const posix_spawn_file_actions_t *,
const posix_spawnattr_t *, char *const,
char *const);

13.4.45 stddef.h

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

13.4.46 stdint.h
#define INT16_C(c)      c
#define INT32_C(c)      c
#define INT8_C(c)       c
#define UINT16_C(c)     c
#define UINT8_C(c)      c
#define UINT32_C(c)     c
#define INT8_MIN        (-128)
#define INT_FAST8_MIN   (-128)
#define INT_LEAST8_MIN  (-128)
#define INT32_MIN       (-2147483647-1)
#define INT_LEAST32_MIN (-2147483647-1)
#define SIG_ATOMIC_MIN  (-2147483647-1)
#define INT16_MIN       (-32767-1)
#define INT_LEAST16_MIN (-32767-1)
#define INT64_MIN       (__INT64_C(9223372036854775807)-1)
#define INTMAX_MIN      (__INT64_C(9223372036854775807)-1)
#define INT_FAST64_MIN  (__INT64_C(9223372036854775807)-1)
#define INT_LEAST64_MIN (__INT64_C(9223372036854775807)-1)
#define WINT_MIN        (0u)
#define INT8_MAX        (127)
#define INT_FAST8_MAX   (127)
#define INT_LEAST8_MAX  (127)
#define INT32_MAX       (2147483647)
#define INT_LEAST32_MAX (2147483647)
#define SIG_ATOMIC_MAX  (2147483647)
#define UINT8_MAX       (255)
#define UINT_FAST8_MAX  (255)
#define UINT_LEAST8_MAX (255)
#define INT16_MAX       (32767)
#define INT_LEAST16_MAX (32767)
#define UINT16_MAX      (65535)
#define INT64_MAX       (__INT64_C(9223372036854775807))
#define INTMAX_MAX      (__INT64_C(9223372036854775807))
#define INT_FAST64_MAX  (__INT64_C(9223372036854775807))
#define INT_LEAST64_MAX (__INT64_C(9223372036854775807))
#define UINT64_MAX      (__UINT64_C(18446744073709551615))
#define UINTMAX_MAX     (__UINT64_C(18446744073709551615))
#define UINT_FAST64_MAX (__UINT64_C(18446744073709551615))
#define UINT_LEAST64_MAX (__UINT64_C(18446744073709551615))

typedef signed char int8_t;
typedef short int16_t;
typedef int int32_t;
typedef unsigned char uint8_t;
typedef unsigned short uint16_t;
typedef unsigned int uint32_t;
typedef signed char int_least8_t;
typedef short int int_least16_t;
typedef int int_least32_t;
typedef unsigned char uint_least8_t;
typedef unsigned short uint_least16_t;
typedef unsigned int uint_least32_t;
typedef signed char int_fast8_t;
typedef unsigned char uint_fast8_t;

13.4.47 stdio.h

#define EOF     (-1)
#define P_tmpdir        "/tmp"
```c
#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

extern char *const _sys_errlist[];
extern void clearerr(FILE *);
extern int fclose(FILE *);
extern FILE *fdopen(int, const char *);
extern int fflush_unlocked(FILE *);
extern int fileno(FILE *);
extern FILE *fopen(const char *, const char *);
extern int fprintf(FILE *, const char *, ...);
extern int fputc(int, FILE *);
extern FILE *freopen(const char *, const char *, FILE *);
extern FILE *freopen64(const char *, const char *, FILE *);
extern int fscanf(FILE *, const char *, ...);
extern int fseek(FILE *, long int, int);
extern int fseeko(FILE *, off_t, int);
extern int fseeko64(FILE *, loff_t, int);
extern int ftello(FILE *);
extern loff_t ftello64(FILE *);
extern int getchar(void);
extern int getchar_unlocked(void);
extern int getw(FILE *);
extern int pclose(FILE *);
extern void perror(const char *);
extern FILE *popen(const char *, const char *);
extern int printf(const char *, ...);
extern int putc_unlocked(int, FILE *);
extern int putchar(int);
extern int putchar_unlocked(int);
extern int putw(int, FILE *);
extern int remove(const char *);
extern void rewind(FILE *);
extern void setbuf(FILE *, char *);
extern int setvbuf(char *, size_t *, const char *, int);
extern void setbuf(FILE *, char *);
extern int sprintf(char *, const char *, ...);
extern int sscanf(const char *, const char *, ...);
extern FILE *stderr;
extern FILE *stdin;
extern FILE *stdout;
extern char *tempnam(const char *, const char *);
extern FILE *tmpfile64(void);
extern FILE *tmpfile(void);
extern char *tmpnam(char *);
extern FILE *vfprintf(FILE *, const char *, va_list);
extern int vprintf(const char *, va_list);
extern int feof(FILE *);
```

extern int ferror(FILE *);
extern int fflush(FILE *);
extern int fgetc(FILE *);
extern int fgetpos(FILE *, fpos_t *);
extern char *fgets(char *, int, FILE *);
extern int fputs(const char *, FILE *);
extern size_t fread(void *, size_t, size_t, FILE *);
extern int fsetpos(FILE *, const fpos_t *);
extern long int ftell(FILE *);
extern size_t fwrite(const void *, size_t, size_t, FILE *);
extern int getc(FILE *);
extern int putc(int, FILE *);
extern int puts(const char *);
extern int setvbuf(FILE *, char *, int, size_t);
extern int snprintf(char *, size_t, const char *, ...);
extern int ungetc(int, FILE *);
extern int vsnprintf(char *, size_t, const char *, va_list);
extern void flockfile(FILE *);
extern int asprintf(char **, const char *, ...);
extern int fgetpos64(FILE *, fpos64_t *);
extern FILE *fopen64(const char *, const char *);
extern int fsetpos64(FILE *, const fpos64_t *);
extern int ftrylockfile(FILE *);
extern void funlockfile(FILE *);
extern int getc_unlocked(FILE *);
extern int setbuffer(FILE *, char *, size_t);
extern int vasprintf(char **, const char *, va_list);
extern int vfprintf(int, const char *, va_list);
extern int vfscanf(FILE *, const char *, va_list);
extern int vscanf(const char *, va_list);
extern int vsscanf(const char *, const char *, va_list);
extern size_t __fpending(FILE *);

13.4.48 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 int quot;
    long int rem;
} ldiv_t;

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

© 2007 Linux Foundation
13 Base Libraries

ISO/IEC 23360 Part 1:2007(E)

```c
} lldiv_t;
extern double __strtod_internal(const char *, char **, int);
extern float __strtof_internal(const char *, char **, int);
extern long int __strtol_internal(const char *, char **, int, int);
extern long double __strtold_internal(const char *, char **, int);
extern long long int __strtoll_internal(const char *, char **, int, int);
extern unsigned long int __strtoul_internal(const char *, char **, int, int);
extern unsigned long long int __strtoull_internal(const char *, char **, int, int);
extern long int a64l(const char *);
extern void abort(void);
extern int abs(int);
extern double atof(const char *);
extern int atoi(const char *);
extern long int atol(const char *);
extern long long int atoll(const char *);
extern void *bsearch(const void *, const void *, size_t, size_t, __compar_fn_t);
extern div_t div(int, int);
extern double drand48(void);
extern char *ecvt(double, int, int *, int *);
extern double erand48(unsigned short);
extern void *free(void *);
extern void *calloc(size_t, size_t);
extern void *malloc(size_t);
extern long int random(void);
extern void *realloc(void *, size_t);
extern char *setstate(char *);
```

96 © 2007 Linux Foundation
extern void srand(unsigned int);
extern void srandom(unsigned int);
extern double strtod(const char *, char **);
extern float strtof(const char *, char **);
extern long int strtol(const char *, char **, int);
extern long double strtold(const char *, char **);
extern long long int strtoull(const char *, char **, int);
extern unsigned long int strtoq(const char *, char **, int);
extern unsigned long long int strtouq(const char *, char **, int);
extern void _Exit(int);
extern size_t __ctype_get_mb_cur_max(void);
extern char **environ;
extern char *realpath(const char *, char *);
extern int setenv(const char *, const char *, int);
extern int unsetenv(const char *);
extern int getloadavg(double, int);
extern int mkstemp64(char *);
extern int posix_memalign(void **, size_t, size_t);
extern int posix_openpt(int);

13.4.49 string.h

#define strerror_r __xpg_strerror_r

extern void *__mempcpy(void *, const void *, size_t);
extern char *__stpcpy(char *, const char *);
extern char *__strdup(const char *);
extern char *__strtok_r(char *, const char *, char **);
extern void *memchr(const void *, int, size_t);
extern int memcmp(const void *, const void *, size_t);
extern void *memcpy(void *, const void *, size_t);
extern void *memmem(const void *, size_t, const void *, size_t);
extern void *memmove(void *, const void *, size_t);
extern void *memset(void *, int, size_t);
extern char *strcat(char *, const char *);
extern char *strchr(const char *, int);
extern char *strerror(int);
extern size_t strlen(const char *);
extern char *strncat(char *, const char *, size_t);
extern char *strncpy(char *, const char *, size_t);
extern char *strpbrk(const char *, const char *);
extern char *strrchr(const char *, int);
extern size_t strspn(const char *, const char *);
extern char *strstr(const char *, const char *);
extern char *strtok(char *, const char *);
extern size_t strxfrm(char *, const char *, size_t);
extern void *memccpy(void *, const void *, int, size_t);
extern char *strdup(const char *);
extern char *strndup(const char *, size_t);
extern size_t strnlen(const char *, size_t);
extern char *strsep(char **, const char *);
extern char *strcasestr(const char *, const char *);
extern char *stpcpy(char *, const char *);
extern char *stpncpy(char *, const char *, size_t);
13 Base Libraries
ISO/IEC 23360 Part 1:2007(E)

extern void *memrchr(const void *, int, size_t);
extern char *__xpg_strerror_r(int, char *, size_t);

13.4.50 strings.h

extern void bcopy(const void *, void *, size_t);
extern int bcmp(const void *, const void *, size_t);
extern void bzero(void *, size_t);
extern int ffs(int);
extern char *index(const char *, int);
extern char *rindex(const char *, int);
extern int strcasecmp(const char *, const char *);
extern int strncasecmp(const char *, const char *, size_t);

13.4.51 sys/file.h

#define LOCK_SH 1
#define LOCK_EX 2
#define LOCK_NB 4
#define LOCK_UN 8

extern int flock(int, int);

13.4.52 sys/ioctl.h

struct winsize {
    unsigned short ws_row;
    unsigned short ws_col;
    unsigned short ws_xpixel;
    unsigned short ws_ypixel;
};
extern int ioctl(int, unsigned long int, ...);

13.4.53 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

extern key_t ftok(const char *, int);

13.4.54 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 MREMAP_MAYMOVE 1
#define MS_ASYNC 1
#define MREMAP_FIXED 2
#define MS_INVALIDATE 2
```c
#define MS_SYNC 4
#define MAP_ANON MAP_ANONYMOUS
extern void *mremap(void *, size_t, size_t, int);
extern int posix_madvise(void *, size_t, int);
extern int msync(void *, size_t, int);
extern int mlock(const void *, size_t);
extern int mlockall(int);
extern void *mmap(void *, size_t, int, int, int, off_t);
extern int mprotect(void *, size_t, int);
extern int munlock(const void *, size_t);
extern int munlockall(void);
extern int munmap(void *, size_t);
extern void *mmap64(void *, size_t, int, int, int, off64_t);
extern int shm_open(const char *, int, mode_t);
extern int shm_unlink(const char *);

13.4.55 sys/msg.h
#define MSG_NOERROR 010000
extern int msgctl(int, int, struct msqid_ds *

13.4.56 sys/param.h
#define NOFILE 256
#define MAXPATHLEN 4096

13.4.57 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 int nfds_t;

13.4.58 sys/resource.h
#define RUSAGE_CHILDREN (-1)
#define RLIM_INFINITY (~0UL)
#define RLIM_SAVED_CUR -1
#define RLIM_SAVING_MAX -1
#define RLIMIT_CPU 0
#define RLIMIT_SELF 0
#define RLIMITFSIZE 1
#define RLIMIT_DATA 2
#define RLIMIT_STACK 3
#define RLIMIT_CORE 4
#define RLIMIT_NOFILE 7
#define RLIMIT_AS 9
```
typedef unsigned long int rlim_t;
typedef unsigned long long int 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 int ru_maxrss;
    long int ru_ixrss;
    long int ru_idrss;
    long int ru_isrss;
    long int ru_minflt;
    long int ru_majflt;
    long int ru_nswap;
    long int ru_inblock;
    long int ru_outblock;
    long int ru_msgsnd;
    long int ru_msgrcv;
    long int ru_nsignals;
    long int ru_nvcsw;
    long int ru_nivcsw;
};

enum __priority_which {
    PRIO_PROCESS = 0,
    PRIO_PGRP = 1,
    PRIO_USER = 2
};
#define PRIO_PGRP       PRIO_PGRP
#define PRIO_PROCESS    PRIO_PROCESS
#define PRIO_USER       PRIO_USER

typedef enum __priority_which __priority_which_t;
typedef enum __priority_which __priority_which_t;
extern int getpriority(__priority_which_t, id_t);
extern int getrlimit64(id_t, struct rlimit64 *);
extern int setpriority(__priority_which_t, id_t, int);
extern int setrlimit(__rlimit_resource_t, const struct rlimit *);
extern int setrlimit64(__rlimit_resource_t, const struct rlimit64 *);
extern int getrlimit(__rlimit_resource_t, struct rlimit *);
extern int getrusage(int, struct rusage *);

13.4.59 sys/select.h

extern int pselect(int, fd_set *, fd_set *, fd_set *,
                   const struct timespec *, const sigset_t *);

13.4.60 sys/sem.h

#define SEM_UNDO 0x1000
#define GETPID 11
#define GETVAL 12
#define GETALL 13

#define GETNCNT 14
#define GETZCNT 15
#define SETVAL  16
#define SETALL  17

struct sembuf {
    short sem_num;
    short sem_op;
    short sem_flg;
};
extern int semctl(int, int, int, ...);
extern int semget(key_t, int, int);
extern int semop(int, struct sembuf *, size_t);

13.4.61 sys/shm.h

#define SHM_RDONLY      010000
#define SHM_W   0200
#define SHM_RND 020000
#define SHM_R   0400
#define SHM_REMAP       040000
#define SHM_LOCK        11
#define SHM_UNLOCK      12

extern int __getpagesize(void);
extern void *shmat(int, const void *, int);
extern int shmctl(int, int, struct shmid_ds *);
extern int shmdt(const void *);
extern int shmget(key_t, size_t, int);

13.4.62 sys/socket.h

#define CMSG_LEN(len)   (CMSG_ALIGN(sizeof(struct cmsghdr))+(len))
#define SCM_RIGHTS      0x01
#define SOL_SOCKET      1
#define SOMAXCONN       128
#define SOL_RAW 255
#define CMSG_ALIGN(len) 
   (sizeof(struct cmsghdr))+CMSG_ALIGN(len)
#define CMSG_DATA(cmsg) 
   ((unsigned char *) (cmsg) + CMSG_ALIGN(sizeof(struct cmsghdr)))
#define CMSG_SPACE(len) 
   (CMSG_ALIGN(sizeof(struct cmsghdr)) + CMSG_ALIGN(len))
#define CMSG_FIRSTHDR(msg)      
   ((msg)->msg_controllen >= sizeof(struct cmsghdr) ? 
    (struct cmsghdr *)(msg)->msg_control : 
    (struct cmsghdr *)NULL)
#define CMSG_NXTHDR(mhdr,cmsg)  
   (((cmsg) == NULL) ? CMSG_FIRSTHDR(mhdr) : 
   (((u_char *)(cmsg) + CMSG_ALIGN((cmsg)->cmsg_len) 
    + (mhdr)-msg_controllen) ? 
    (struct cmsghdr *)(msg)->msg_control : 
    (struct cmsghdr *)(u_char *)(msg) + CMSG_ALIGN((msg)->cmsg_len)))

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_align __ss_align;
    char __ss_padding[128 - (2 * sizeof(__ss_align))];
};

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 PF_UNSPEC       AF_UNSPEC

#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 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 SO_TYPE 3
#define SO_ACCEPTCONN   30
#define SO_ERROR        4
#define SO_DONTROUTE    5
#define SO_BROADCAST    6
#define SO_SNDBUF       7
#define SO_RCVBUF       8
#define SO_KEEPALIVE    9
#define SIOCGIFNAME     0x8910
#define SIOCGIFCONF 0x8912
#define SIOCGIFFLAGS 0x8913
#define SIOCGIFADDR 0x8915
#define SIOCGIFBROADCAST 0x8917
#define SIOCGIFBRDADDR 0x8919
#define SIOCGIFNETMASK 0x891b
#define SIOCGIFMTU 0x8921
#define SIOCGIFHWADDR 0x8927

#define SHUT_RD 0
#define SHUT_WR 1
#define SHUT_RDWR 2
#define MSG_WAITALL 0x100
#define MSG_TRUNC 0x20
#define MSG_EOR 0x80
#define MSG_OOB 1
#define MSG_PEEK 2
#define MSG_DONTROUTE 4
#define MSG_CTRUNC 8

extern int bind(int, const struct sockaddr *, socklen_t);
extern int getnameinfo(const struct sockaddr *, socklen_t, char *,
socklen_t, char *, socklen_t, unsigned int);
extern int getsockname(int, struct sockaddr *, socklen_t *);
extern int listen(int, int);
extern int setsockopt(int, int, int, const void *, socklen_t);
extern int accept(int, struct sockaddr *, socklen_t *);
extern int connect(int, const struct sockaddr *, socklen_t);
extern ssize_t recv(int, void *, size_t, int);
extern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *,
socklen_t *);
extern ssize_t recvmsg(int, struct msghdr *, int);
extern ssize_t send(int, const void *, size_t, int);
extern ssize_t sendmsg(int, const struct msghdr *, int);
extern ssize_t sendto(int, const void *, size_t, int,
const struct sockaddr *, socklen_t);
extern int getpeername(int, struct sockaddr *, socklen_t *);
extern int getsockopt(int, int, int, void *, socklen_t *);
extern int shutdown(int, int);
extern int socket(int, int, int);
extern int socketpair(int, int, int, int);
extern int sockatmark(int);

13.4.63 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_IROWXG (S_IWXG>>3)
#define S_IWOTH (S_IWGRP>>3)
#define S_IRWXG (S_IRWXG>>3)
#define S_IWOTH (S_IWGRP>>3)

© 2007 Linux Foundation


#define S_IWGRP (S_IWUSR>>3)
#define S_IXOTH (S_IXGRP>>3)
#define S_IXGRP (S_IXUSR>>3)
#define S_ISVTX 01000
#define S_IUXUSR 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_IEXEC S_IXUSR
#define S_IRUSR S_IREAD
#define S_IWUSR S_IWRITE
#define S_IXUSR S_IEXEC

extern int __fxstat(int, int, struct stat *);
extern int __fxstat64(int, int, struct stat64 *);
extern int __lxstat(int, const char *, struct stat *);
extern int __lxstat64(int, const char *, struct stat64 *);
extern int __xstat(int, const char *, struct stat *);
extern int __xstat64(int, const char *, struct stat64 *);
extern int mkfifo(const char *, mode_t);
extern int chmod(const char *, mode_t);
extern int fchmod(int, mode_t);
extern mode_t umask(mode_t);

## 13.4.64 sys/statfs.h

#define NFS_SUPER_MAGIC 0x6969

extern int fstatfs64(int, struct statfs64 *);
extern int fstatfs64(const char *, struct statfs64 *);
extern int fstatfs(int, struct statfs *);
extern int fstatfs(const char *, struct statfs *);

## 13.4.65 sys/statvfs.h

extern int fstatvfs(int, struct statvfs *);
extern int fstatvfs64(int, struct statvfs64 *);
extern int fstatvfs64(const char *, struct statvfs64 *);
extern int statvfs64(const char *, struct statvfs64 *);

## 13.4.66 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 int tv_nsec;
};
struct timeval {
    time_t tv_sec;
    suseconds_t tv_usec;
};
struct itimerval {
    struct timeval it_interval;
    struct timeval it_value;
};
extern int getitimer(__itimer_which_t, struct itimerval *);
extern int setitimer(__itimer_which_t, const struct itimerval *,
    struct itimerval *);
extern int adjtime(const struct timeval *, struct timeval *);
extern int gettimeofday(struct timeval *, struct timezone *);
extern int utimes(const char *, const struct timeval *);

13.4.67 sys/timeb.h

struct timeb {
    time_t time;
    unsigned short millitm;
    short timezone;
    short dstflag;
};
extern int ftime(struct timeb *);

13.4.68 sys/times.h

struct tms {
    clock_t tms_utime;
    clock_t tms_stime;
    clock_t tms_cutime;
    clock_t tms_cstime;
};
extern clock_t times(struct tms *);

13.4.69 sys/types.h

#ifndef FALSE
#define FALSE   0
#endif
#ifndef TRUE
#define TRUE    1
#endif
#define FD_SETSIZE      1024
#define FD_ZERO(fdsetp) bzero(fdsetp, sizeof(*(fdsetp)))
#define FD_ISSET(d,set)  
    ((set)-
>fds_bits[((d)/(8*sizeof(long)))]&((1<<((d)%(8*sizeof(long)))))
#define FD_CLR(d,set)   
    ((set)-
>fds_bits[((d)/(8*sizeof(long)))]&=~(1<<((d)%(8*sizeof(long)))))
#define FD_SET(d,set)   

typedef unsigned char u_int8_t;
typedef unsigned short u_int16_t;
typedef unsigned int u_int32_t;
typedef unsigned long long int u_int64_t;
typedef unsigned int uid_t;
typedef int pid_t;
typedef long int off_t;
typedef int key_t;
typedef long int suseconds_t;
typedef unsigned int u_int;
typedef struct {
    int __val[2];
} fsid_t;
typedef unsigned int useconds_t;
typedef long int blksize_t;
typedef long int fd_mask;
typedef void *timer_t;
typedef int clockid_t;
typedef unsigned int id_t;
typedef unsigned long long int ino64_t;
typedef long long int loff_t;
typedef long int blkcnt_t;
typedef unsigned long int fsblkcnt_t;
typedef unsigned long int fsfilcnt_t;
typedef long int blkcnt64_t;
typedef unsigned long long int fsblkcnt64_t;
typedef unsigned long int fsfilcnt64_t;
typedef unsigned char u_char;
typedef unsigned short u_short;
typedef unsigned long int u_long;
typedef unsigned long int ino_t;
typedef unsigned int gid_t;
typedef unsigned long int dev_t;
typedef unsigned int mode_t;
typedef unsigned long int nlink_t;
typedef char *caddr_t;
typedef struct {
    unsigned long int fds_bits[(__FDSET_LONGS)];
} fd_set;
typedef long int clock_t;
typedef long int time_t;

13.4.70 sys/uio.h

extern ssize_t readv(int, const struct iovec *, int);
extern ssize_t writev(int, const struct iovec *, int);

13.4.71 sys/un.h

#define UNIX_PATH_MAX 108

struct sockaddr_un {
    sa_family_t sun_family;
    char sun_path[UNIX_PATH_MAX];
};
13.4.72 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];
};

extern int uname(struct utsname *);

13.4.73 sys/wait.h

#define WIFSIGNALED(status)     (!WIFSTOPPED(status) &&
 WEXITED(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 0x080
#define WSTOPSIG(status)        WEXITSTATUS(status)

typedef enum {
    P_ALL,
    P_PID,
    P_PGID
} idtype_t;

extern int waitid(idtype_t, id_t, siginfo_t *, int);
extern pid_t wait(int *);
extern pid_t waitpid(pid_t, int *, int);
extern pid_t wait4(pid_t, int *, int, struct rusage *);

13.4.74 syslog.h

#define LOG_EMERG     0
#define LOG_PRIMASK   0x07
#define LOG_ALERT     1
#define LOG_CRIT      2
#define LOG_ERR       3
#define LOG_WARNING   4
#define LOG_NOTICE    5
#define LOG_INFO      6
#define LOG_DEBUG     7
#define LOG_KERN      (0<<3)
#define LOG_AUTHPRIV  (10<<3)
#define LOG_FTP       (11<<3)
#define LOG_USER      (1<<3)
#define LOG_MAIL      (2<<3)
#define LOG_DAEMON    (3<<3)
#define LOG_AUTH      (4<<3)
#define LOG_SYSLOG    (5<<3)
#define LOG_LPR       (6<<3)
#define LOG_NEWS      (7<<3)
#define LOG_UUCP      (8<<3)
#define LOG_CRON      (9<<3)
13 Base Libraries

ISO/IEC 23360 Part 1:2007(E)

#define LOG_FACMASK 0x03f8
#define LOG_LOCAL0 (16<<3)
#define LOG_LOCAL1 (17<<3)
#define LOG_LOCAL2 (18<<3)
#define LOG_LOCAL3 (19<<3)
#define LOG_LOCAL4 (20<<3)
#define LOG_LOCAL5 (21<<3)
#define LOG_LOCAL6 (22<<3)
#define LOG_LOCAL7 (23<<3)
#define LOG_UPTO(pri) ((1 << ((pri)+1)) - 1)
#define LOG_MASK(pri) (1 << (pri))
#define LOG_PID 0x01
#define LOG_CONS 0x02
#define LOG_ODELAY 0x04
#define LOG_NDELAY 0x08
#define LOG_NOWAIT 0x10
#define LOG_PERROR 0x20
extern void closelog(void);
extern void openlog(const char *, int, int);
extern int setlogmask(int);
extern void syslog(int, const char *, ...);
extern void vsyslog(int, const char *, va_list);

13.4.75 termios.h

#define TCIflush 0
#define TCOOFF 0
#define TCSANOW 0
#define BS0 0000000
#define CR0 0000000
#define FF0 0000000
#define NL0 0000000
#define TAB0 0000000
#define VT0 0000000
#define OPOST 0000001
#define OCRNL 0000010
#define ONOCR 0000020
#define ONLRET 0000040
#define OFILL 0000100
#define OFDEL 0000200
#define NL1 0000400
#define TCOFLUSH 1
#define TCOON 1
#define TCSADRAIN 1
#define TCIoff 2
#define TCIoflush 2
#define TCSAflush 2
#define TCION 3

typedef unsigned int speed_t;
typedef unsigned char cc_t;
typedef unsigned int tcflag_t;

#define NCCS 32

struct termios { tcc_t c_iflag; tcc_t c_oflag; tcc_t c_cflag; tcc_t c_lflag; cc_t c_line;};
ISO/IEC 23360 Part 1:2007(E)

13 Base Libraries

cc_t c_cc[NCCS];
speed_t c_ispeed;
speed_t c_ospeed;
};

#define VINTR 0
#define VQUIT 1
#define VLNEXT 15
#define VERASE 2
#define VKILL 3
#define VEOF 4

#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

extern speed_t cfgetispeed(const struct termios *);
extern speed_t cfgetospeed(const struct termios *);
extern void cfmakeraw(struct termios *);
extern int cfsetispeed(struct termios *, speed_t);
extern int cfsetospeed(struct termios *, speed_t);
extern int cfsetspeed(struct termios *, speed_t);
extern int tcflow(int, int);
extern int tcflush(int, int);
extern int tcsetattr(int, int, const struct termios *);
extern int tcdrain(int);
extern int tcgetattr(int, struct termios *);

13.4.76 time.h

#define CLK_TCK ((__clock_t)__sysconf(2))
#define CLOCK_REALTIME 0
#define TIMER_ABSTIME 1
#define CLOCKS_PER_SEC 10000001
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 int tm_gmtoff;
    char *tm_zone;
};

struct itimerspec {
    struct timespec it_interval;
    struct timespec it_value;
};

extern int __daylight;
extern long int __timezone;
extern char *__tzname[];
extern char *asctime(const struct tm *);
extern clock_t clock(void);
extern char *ctime(const time_t *);
extern char *ctime_r(const time_t *, char *);
extern double difftime(time_t, time_t);
extern struct tm *getdate(const char *);
extern int getdate_err;
extern struct tm *gmtime(const time_t *);
extern struct tm *localtime(const time_t *);
extern time_t mktime(struct tm *);
extern int stime(const time_t *);
extern size_t strftime(char *, size_t, const char *, const struct tm *);
extern char *strptime(const char *, const char *, struct tm *);
extern time_t time(time_t *);
extern int nanosleep(const struct timespec *, struct timespec *);
extern int daylight;
extern long int timezone;
extern char *tzname[];
extern void tzset(void);
extern char *asctime_r(const struct tm *, char *);
extern struct tm *gmtime_r(const time_t *, struct tm *);
extern struct tm *localtime_r(const time_t *, struct tm *);
extern int clock_getcpuclockid(pid_t, clockid_t *);
extern int clock_getres(clockid_t, struct timespec *);
extern int clock_gettime(clockid_t, struct timespec *);
extern int clock_nanosleep(clockid_t, int, const struct timespec *,
    struct timespec *);
extern int clock_settime(clockid_t, const struct timespec *);
extern int timer_create(clockid_t, struct sigevent *, timer_t *);
extern int timer_delete(timer_t);
extern int timer_getoverrun(timer_t);
extern int timer_gettime(timer_t, struct itimerspec *);
extern int timer_settime(timer_t, int, const struct itimerspec *,
    struct itimerspec *);

13.4.77 ucontext.h

extern int getcontext(ucontext_t *);
extern void makecontext(ucontext_t *, void (*)(void), int, ...);
extern int setcontext(const struct ucontext *);
extern int swapcontext(ucontext_t *, const struct ucontext *);

13.4.78 ulimit.h

#define UL_GETFSIZE 1
#define UL_SETFSIZE 2

extern long int ulimit(int, ...);

13.4.79 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 int 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_SEMAPHORES 200112
#define _POSIX_SHARED_MEMORY_OBJECTS 200112
#define _POSIX_TIMERS 200112
#define _PC_LINK_MAX 0
#define _PC_MAX_CANON 1
#define _PC_ASYNC_IO 10
#define _PC_PRIO_IO 11
#define _PC_FILESIZEBITS 13
#define _PC_REC_INCR_XFER_SIZE 14
#define _PC_REC_MIN_XFER_SIZE 16
#define _PC_REC_XFER_ALIGN 17
#define _PC_ALLOC_SIZE_MIN 18
#define _PC_MAX_INPUT 2
#define _PC_2_SYMLINKS 20
#define _PC_NAME_MAX 3
#define _PC_PATH_MAX 4
#define _PC_PIPE_BUF 5
#define _PC_CHOWN_RESTRICTED 6
#define _PC_NO_TRUNC 7
#define _PC_VDISABLE 8
#define _PC_SYNC_IO 9

#define _SC_ARG_MAX 0
#define _SC_CHILD_MAX 1
#define _SC_PRIORITY_SCHEDULING 10
#define _SC_XOPEN_XPG4 100
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_SC_CHAR_BIT</td>
<td>101</td>
</tr>
<tr>
<td>_SC_CHAR_MAX</td>
<td>102</td>
</tr>
<tr>
<td>_SC_CHAR_MIN</td>
<td>103</td>
</tr>
<tr>
<td>_SC_INT_MAX</td>
<td>104</td>
</tr>
<tr>
<td>_SC_INT_MIN</td>
<td>105</td>
</tr>
<tr>
<td>_SC_LONG_BIT</td>
<td>106</td>
</tr>
<tr>
<td>_SC_WORD_BIT</td>
<td>107</td>
</tr>
<tr>
<td>_SC_MB_LEN_MAX</td>
<td>108</td>
</tr>
<tr>
<td>_SC_NZERO</td>
<td>109</td>
</tr>
<tr>
<td>_SC_TIMERS</td>
<td>11</td>
</tr>
<tr>
<td>_SC_SSIZE_MAX</td>
<td>110</td>
</tr>
<tr>
<td>_SC_SCHAR_MAX</td>
<td>111</td>
</tr>
<tr>
<td>_SC_SCHAR_MIN</td>
<td>112</td>
</tr>
<tr>
<td>_SC_SHRT_MAX</td>
<td>113</td>
</tr>
<tr>
<td>_SC_SHRT_MIN</td>
<td>114</td>
</tr>
<tr>
<td>_SC_UCHAR_MAX</td>
<td>115</td>
</tr>
<tr>
<td>_SC_UINT_MAX</td>
<td>116</td>
</tr>
<tr>
<td>_SC_ULONG_MAX</td>
<td>117</td>
</tr>
<tr>
<td>_SC_USHRT_MAX</td>
<td>118</td>
</tr>
<tr>
<td>_SC_NL_ARGMAX</td>
<td>119</td>
</tr>
<tr>
<td>_SC_ASYNCHRONOUS_IO</td>
<td>12</td>
</tr>
<tr>
<td>_SC_NL_LANGMAX</td>
<td>120</td>
</tr>
<tr>
<td>_SC_NL_MSGMAX</td>
<td>121</td>
</tr>
<tr>
<td>_SC_NL_NMAX</td>
<td>122</td>
</tr>
<tr>
<td>_SC_NL_SETMAX</td>
<td>123</td>
</tr>
<tr>
<td>_SC_NL_TEXTMAX</td>
<td>124</td>
</tr>
<tr>
<td>_SC_XBS5_ILP32_OFFSET</td>
<td>125</td>
</tr>
<tr>
<td>_SC_XBS5_ILP32_OFFSET_BIG</td>
<td>126</td>
</tr>
<tr>
<td>_SC_XBS5_LP64_OFFSET</td>
<td>127</td>
</tr>
<tr>
<td>_SC_XBS5_LPBIG_OFFSET_BIG</td>
<td>128</td>
</tr>
<tr>
<td>_SC_XOPEN_LEGACY</td>
<td>129</td>
</tr>
<tr>
<td>_SC_PRIORITIZED_IO</td>
<td>13</td>
</tr>
<tr>
<td>_SC_XOPEN_REALTIME</td>
<td>130</td>
</tr>
<tr>
<td>_SC_XOPEN_REALTIME_THREADS</td>
<td>131</td>
</tr>
<tr>
<td>_SC_ADVISORY_INFO</td>
<td>132</td>
</tr>
<tr>
<td>_SC_BARRIERS</td>
<td>133</td>
</tr>
<tr>
<td>_SC_BASE</td>
<td>134</td>
</tr>
<tr>
<td>_SC_C_LANG_SUPPORT</td>
<td>135</td>
</tr>
<tr>
<td>_SC_C_LANG_SUPPORT_R</td>
<td>136</td>
</tr>
<tr>
<td>_SC_CLOCK_SELECTION</td>
<td>137</td>
</tr>
<tr>
<td>_SC_CPUTIME</td>
<td>138</td>
</tr>
<tr>
<td>_SC_THREAD_CPUTIME</td>
<td>139</td>
</tr>
<tr>
<td>_SC_SONYRONIZED_IO</td>
<td>14</td>
</tr>
<tr>
<td>_SC_DEVICE_IO</td>
<td>140</td>
</tr>
<tr>
<td>_SC_DEVICE_SPECIFIC</td>
<td>141</td>
</tr>
<tr>
<td>_SC_DEVICE_SPECIFIC_R</td>
<td>142</td>
</tr>
<tr>
<td>_SC_FD_MGMT</td>
<td>143</td>
</tr>
<tr>
<td>_SC_FIFO</td>
<td>144</td>
</tr>
<tr>
<td>_SC_PIPE</td>
<td>145</td>
</tr>
<tr>
<td>_SC_FILE_ATTRIBUTES</td>
<td>146</td>
</tr>
<tr>
<td>_SC_FILE_LOCKING</td>
<td>147</td>
</tr>
<tr>
<td>_SC_FILE_SYSTEM</td>
<td>148</td>
</tr>
<tr>
<td>_SC_MONOTONIC_CLOCK</td>
<td>149</td>
</tr>
<tr>
<td>_SC_FSYNC</td>
<td>15</td>
</tr>
<tr>
<td>_SC_MULTI_PROCESS</td>
<td>150</td>
</tr>
<tr>
<td>_SC_SINGLE_PROCESS</td>
<td>151</td>
</tr>
<tr>
<td>_SC_NETWORKING</td>
<td>152</td>
</tr>
<tr>
<td>_SC_READER_WRITER_LOCKS</td>
<td>153</td>
</tr>
<tr>
<td>_SC_SPIN_LOCKS</td>
<td>154</td>
</tr>
<tr>
<td>_SC_REGEXP</td>
<td>155</td>
</tr>
<tr>
<td>_SC_REGEX_VERSION</td>
<td>156</td>
</tr>
<tr>
<td>_SC_SHELL</td>
<td>157</td>
</tr>
<tr>
<td>_SC_SIGNALS</td>
<td>158</td>
</tr>
<tr>
<td>_SC_SPAWN</td>
<td>159</td>
</tr>
<tr>
<td>_SC_MAPPED_FILES</td>
<td>16</td>
</tr>
<tr>
<td>_SC_SPORADIC_SERVER</td>
<td>160</td>
</tr>
</tbody>
</table>
#define _SC_THREAD_SPORADIC_SERVER 161
#define _SC_SYSTEM_DATABASE 162
#define _SC_SYSTEM_DATABASE_R 163
#define _SC_TIMESTAMPS 164
#define _SC_TYPED_MEMORY_OBJECTS 165
#define _SC_USER_GROUPS 166
#define _SC_USER_GROUPS_R 167
#define _SC_2_PBS 168
#define _SC_2_PBS_ACCOUNTING 169
#define _SC_MEMLOCK 17
#define _SC_2_PBS_LOCATE 170
#define _SC_2_PBS_MESSAGE 171
#define _SC_2_PBS_TRACK 172
#define _SC_SYSCOMPLETE_MAX 173
#define _SC_STREAMS 174
#define _SC_2_PBS_CHECKPOINT 175
#define _SC_V6_ILP32_OFF32 176
#define _SC_V6_ILP32_OFFBIG 177
#define _SC_V6_LP64_OFF64 178
#define _SC_V6_LP64_OFFBIG 179
#define _SC_MEMLOCK_RANGE 18
#define _SC_HOST_NAME_MAX 180
#define _SC_TRACE 181
#define _SC_TRACE_EVENT_FILTER 182
#define _SC_TRACE_INHERIT 183
#define _SC_TRACE_LOG 184
#define _SC_LEVEL1_ICACHE_SIZE 185
#define _SC_LEVEL1_ICACHE_ASSOC 186
#define _SC_LEVEL1_ICACHE_LINESIZE 187
#define _SC_LEVEL1_DCACHE_SIZE 188
#define _SC_LEVEL1_DCACHE_ASSOC 189
#define _SC_MEMORY_PROTECTION 19
#define _SC_LEVEL1_DCACHE_LINESIZE 190
#define _SC_LEVEL2_CACHE_SIZE 191
#define _SC_LEVEL2_CACHE_ASSOC 192
#define _SC_LEVEL2_CACHE_LINESIZE 193
#define _SC_LEVEL3_CACHE_SIZE 194
#define _SC_LEVEL3_CACHE_ASSOC 195
#define _SC_LEVEL3_CACHE_LINESIZE 196
#define _SC_LEVEL4_CACHE_SIZE 197
#define _SC_LEVEL4_CACHE_ASSOC 198
#define _SC_LEVEL4_CACHE_LINESIZE 199
#define _SC_CLK_TCK 2
#define _SC_MESSAGE_PASSING 20
#define _SC_SEMaphores 21
#define _SC_SHARED_MEMORY_OBJECTS 22
#define _SC_AIO_LISTIO_MAX 23
#define _SC_IPV6 235
#define _SC_RAW_SOCKETS 236
#define _SC_AIO_MAX 24
#define _SC_AIO_PRIO_DELTA_MAX 25
#define _SC_DELAYTIMER_MAX 26
#define _SC_MQ_OPEN_MAX 27
#define _SC_MQ_PRIO_MAX 28
#define _SC_VERSION 29
#define _SC_MQ_OPEN_MAX 30
#define _SC_PAGESIZE 30
#define _SC_PAGE_SIZE 30
#define _SC_RT-navbar: 31
#define _SC_SEM_NSEMS_MAX 32
#define _SC_SEM_VALUE_MAX 33
#define _SC_SIGQUEUE_MAX 34
#define _SC_TIMERS_MAX 35
#define _SC_BC_BASE_MAX 36
#define _SC_BC_DIM_MAX 37
#define _SC_BC_SCALE_MAX 38

© 2007 Linux Foundation

113
#define _SC_BC_STRING_MAX 39
#define _SC_OPEN_MAX 4
#define _SC_COLL_WEIGHTS_MAX 40
#define _SC_EQUIV_CLASS_MAX 41
#define _SC_EXPR_NEST_MAX 42
#define _SC_LINE_MAX 43
#define _SC_RE_DUP_MAX 44
#define _SC_CHARCLASS_NAME_MAX 45
#define _SC_2_VERSION 46
#define _SC_2_C_BIND 47
#define _SC_2_C_DEV 48
#define _SC_2_FORT_DEV 49
#define _SC_STREAM_MAX 5
#define _SC_2_FORT_RUN 50
#define _SC_2_SW_DEV 51
#define _SC_2_LOCALEDEF 52
#define _SC_PII 53
#define _SC_PII_XTI 54
#define _SC_PII_SOCKET 55
#define _SC_PII_INTERNET 56
#define _SC_PII_OSI 57
#define _SC_PII_CLTS 58
#define _SC_PII_INTERNET_STREAM 59
#define _SC_PII_INTERNET_DGRAM 60
#define _SC_PII_OSI_COTS 61
#define _SC_PII_OSI_CLTS 62
#define _SC_THREADS 63
#define _SC_THREAD_SAFE_FUNCTIONS 64
#define _SC_GETGR_R_SIZE_MAX 65
#define _SC_JOB_CONTROL 66
#define _SC_GETPW_R_SIZE_MAX 67
#define _SC_LOGIN_NAME_MAX 68
#define _SC_TTY_NAME_MAX 69
#define _SC_THREAD_DESTRUCTOR_ITERATIONS 70
#define _SC_THREAD_STACK_MIN 71
#define _SC_THREAD_STACK_MAX 72
#define _SC_THREAD_ATTR_STACKADDR 73
#define _SC_THREAD_ATTR_STACKSIZE 74
#define _SC_THREAD_PRIORITY_SCHEDULING 75
#define _SC_SAVED_IDS 76
#define _SC_THREAD_PRIO_INHERIT 77
#define _SC_THREAD_PRIO_PROTECT 78
#define _SC_THREAD_PROCESS_SHARED 79
#define _SC_NPROCESSORS_CONF 80
#define _SC_NPROCESSORS_ONLN 81
#define _SC_PHYS_PAGES 82
#define _SC_AVPHYS_PAGES 83
#define _SC_ATEXIT_MAX 84
#define _SC_PASS_MAX 85
#define _SC_XOPEN_VERSION 86
#define _SC_REALTIME_SIGNALS 87
#define _SC_XOPEN_XCU_VERSION 88
#define _SC_XOPEN_UNIX 89
#define _SC_XOPEN_CRYPT 90
#define _SC_XOPEN_ENH_I18N 91
#define _SC_XOPEN_SHM 92
#define _SC_2_CHAR_TERM 93
#define _SC_2_C_VERSION 94
#define _SC_2_UPE 95
#define _SC_2_VERSION 96
#define _SC_XOPEN_XPG2  98
#define _SC_XOPEN_XPG3  99
#define _CS_PATH        0
#define _POSIX_REGEXP   1
#define _CS_XBS5_ILP32_OFF32_CFLAGS     1100
#define _CS_XBS5_ILP32_OFF32_LDFLAGS    1101
#define _CS_XBS5_ILP32_OFF32_LIBS       1102
#define _CS_XBS5_ILP32_OFF32_LINTFLAGS  1103
#define _CS_XBS5_ILP32_OFFBIG_CFLAGS    1104
#define _CS_XBS5_ILP32_OFFBIG_LDFLAGS   1105
#define _CS_XBS5_ILP32_OFFBIG_LIBS      1106
#define _CS_XBS5_ILP32_OFFBIG_LINTFLAGS 1107
#define _CS_XBS5_LP64_OFF64_CFLAGS      1108
#define _CS_XBS5_LP64_OFF64_LDFLAGS     1109
#define _CS_XBS5_LP64_OFF64_LIBS        1110
#define _CS_XBS5_LP64_OFF64_LINTFLAGS   1111
#define _CS_XBS5_LPBIG_OFFBIG_CFLAGS    1112
#define _CS_XBS5_LPBIG_OFFBIG_LDFLAGS   1113
#define _CS_XBS5_LPBIG_OFFBIG_LIBS      1114
#define _CS_XBS5_LPBIG_OFFBIG_LINTFLAGS 1115
#define _XOPEN_XPG4     1
#define F_ULOCK 0
#define F_LOCK  1
#define F_TLOCK 2
#define F_TEST  3

extern int getdtablesize(void);
extern char **__environ;
extern pid_t __getpgid(pid_t);
extern void _exit(int);
extern int acct(const char *);
extern unsigned int alarm(unsigned int);
extern int chown(const char *, uid_t, gid_t);
extern int chroot(const char *);
extern size_t confstr(int, char *, size_t);
extern char *ctermid(char *);
extern char *cuserid(char *);
extern int daemon(int, int);
extern int execl(const char *, const char *, ...);
extern int execle(const char *, const char *, ...);
extern int execlp(const char *, const char *, ...);
extern int execv(const char *, char *const);
extern int execvp(const char *, char *const);
extern int fdatasync(int);
extern int ftruncate64(int, off64_t);
extern int getdomainname(char *, size_t);
extern long int gethostid(void);
extern char *getlogin(void);
extern int getlogin_r(char *, size_t);
extern int getopt(int, char *const, const char *);
extern pid_t getpgid(void);
extern pid_t getsid(pid_t);
extern char *getwd(char *);
extern int lockf(int, int, off_t);
extern int lockf64(int, int, off_t);
extern int mkstemp(char *);
extern int nice(int);
extern char *optarg;
extern int opterr;
extern int optind;
extern int optopt;
extern int rename(const char *, const char *);
extern int setegid(gid_t);
extern int seteuid(uid_t);
extern int sethostname(const char *, size_t);
extern int setpgid(void);
extern void swab(const void *, void *, ssize_t);
extern void sync(void);
extern pid_t tcgetpgrp(int);
extern int tcsetpgrp(int, pid_t);
extern int truncate(const char *, off_t);
extern int truncate64(const char *, off64_t);
extern char *ttyname(int);
extern unsigned int ualarm(useconds_t, useconds_t);
extern int usleep(useconds_t);
extern int close(int);
extern int fsync(int);
extern off_t lseek(int, off_t, int);
extern int pause(void);
extern ssize_t read(int, void *, size_t);
extern char *crypt(const char *, const char *);
extern void encrypt(char *, int);
extern void setkey(const char *);
extern int access(const char *, int);
extern int brk(void *);
extern int chmod(const char *, mode_t);
extern pid_t chdir(const char *);
extern int dup(int);
extern int dup2(int, int);
extern int execve(const char *, char *const, char *const);
extern int fchdir(int);
extern int fchown(int, uid_t, gid_t);
extern int fork(void);
extern gid_t getegid(void);
extern uid_t geteuid(void);
extern gid_t getgid(void);
extern int getgroups(int, gid_t);
extern int gethostname(char *, size_t);
extern pid_t getpgrp(pid_t);
extern pid_t getpid(void);
extern uid_t getuid(void);
extern int lchown(const char *, uid_t, gid_t);
extern int link(const char *, mode_t);
extern int lseek(int, off_t, int);
extern int pipe(int);
extern ssize_t readlink(const char *, char *, size_t);
extern int mkdir(const char *, mode_t);
extern long int pathconf(const char *, int);
extern int pipe(int);
extern ssize_t pread(int, void *, size_t, off_t);
extern ssize_t pwrite(int, const void *, size_t, off_t);
extern char **_environ;
extern long int fpathconf(int, int);
extern int ftruncate(int, off_t);
extern int getpagesize(void);
extern pid_t getppid(void);
extern int lsatty(int);
extern loff_t lseek64(int, loff_t, int);
extern ssize_t pread64(int, void *, size_t, off64_t);
extern ssize_t pwrite64(int, const void *, size_t, off64_t);
extern int ttyname_r(int, char *, size_t);

13.4.80 utime.h

struct utimbuf {
    time_t actime;
    time_t modtime;
};
extern int utime(const char *, const struct utimbuf *);

13.4.81 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 DEAD_PROCESS 8
#define ACCOUNTING 9

extern void endutent(void);
extern struct utmp *getutent(void);
extern void setutent(void);
extern int getutent_r(struct utmp *, struct utmp **);
extern int utmpname(const char *);
extern int login_tty(int);
extern void login(const struct utmp *);
extern int logout(const char *);
extern void logwtmp(const char *, const char *, const char *);

13.4.82 utmpx.h

extern void endutxent(void);
extern struct utmpx *getutxent(void);
extern void setutxent(void);
extern int getutxent_r(struct utmp *, struct utmpx **);
extern int getutxid(const struct utmpx *);
extern int utmpname(const char *);
extern int login_tty(int);
extern int logout(const char *);
extern void logwtmp(const char *, const char *, const char *, const char *);

13.4.83 wchar.h

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

© 2007 Linux Foundation
extern double __wcstod_internal(const wchar_t *, wchar_t * *, int);
extern float __wcstof_internal(const wchar_t *, wchar_t * *, int);
extern long int __wcstol_internal(const wchar_t *, wchar_t * *, int, int);
extern long double __wcstold_internal(const wchar_t *, wchar_t * *, int, int);
extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t * *
int, int);
extern wchar_t *wcscat(wchar_t *, const wchar_t *);
extern wchar_t *wcschr(const wchar_t *, wchar_t);
extern int wcscmp(const wchar_t *, const wchar_t *);
extern int wcscoll(const wchar_t *, const wchar_t *);
extern wchar_t *wcscpy(wchar_t *, const wchar_t *);
extern size_t wcscspn(const wchar_t *, const wchar_t *);
extern wchar_t *wcsdup(const wchar_t *);
extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wcspbrk(const wchar_t *, const wchar_t *);
extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
extern size_t wcsspn(const wchar_t *, const wchar_t *);
extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t * *);
extern int wcswidth(const wchar_t *, size_t);
extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
extern int wctob(wint_t);
extern int wcwidth(wchar_t);
extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
extern size_t mbrlen(const char *, size_t, mbstate_t *);
extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
extern int mbsinit(const mbstate_t *);
extern size_t mbstowcs(wchar_t *, const char *, size_t, size_t, mbstate_t *);
extern size_t mbstowcs(wchar_t *, const char *, size_t, mbstate_t *);
extern size_t mbstowcs(wchar_t *, const char *, size_t, mbstate_t *);
extern size_t wcsrtombs(char *, const wchar_t *, size_t, mbstate_t *);
extern size_t wcsrtombs(char *, const wchar_t *, size_t, mbstate_t *);
extern size_t wcsrtombs(char *, const wchar_t *, size_t, mbstate_t *);
extern double wcstod(const wchar_t *, wchar_t * *);
extern float wcstof(const wchar_t *, wchar_t * *);
extern long int wcstol(const wchar_t *, wchar_t * *, int);
extern long double wcstold(const wchar_t *, wchar_t * *, int);
extern unsigned long int wcstoul(const wchar_t *, wchar_t * *, int);
extern unsigned long int wcstouq(const wchar_t *, wchar_t * *, int);
extern wchar_t *wcsxcs(const wchar_t *, const wchar_t *);
extern int wcsscasecmp(const wchar_t *, const wchar_t *);
extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
13 Base Libraries

extern size_t wcslen(const wchar_t *, size_t);
extern long long int wcstoll(const wchar_t *, wchar_t **, int);
extern unsigned long long int wcstoull(const wchar_t *, wchar_t **,
int);
extern wint_t btowc(int);
extern wint_t fgetwc(FILE *);
extern wint_t fgetwc_unlocked(FILE *);
extern wchar_t *fgetws(wchar_t *, int, FILE *);
extern wint_t fputwc(wchar_t, FILE *);
extern int fputc(wchar_t, FILE *);
extern wchar_t *fgetws(wchar_t *, int, FILE *);
extern wint_t getwc(FILE *);
extern wint_t getwchar(void);
extern wint_t putwc(wchar_t, FILE *);
extern wint_t putwchar(wchar_t);
extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
extern int swscanf(const wchar_t *, const wchar_t *, ...);
extern wint_t ungetwc(wint_t, FILE *);
extern int vfwprintf(FILE *, const wchar_t *, va_list);
extern int vfwscanf(FILE *, const wchar_t *, va_list);
extern int vswprintf(wchar_t *, size_t, const wchar_t *,
va_list);
extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
extern int wprintf(const wchar_t *, ...);
extern int wscanf(const wchar_t *, ...);

13.4.84 wctype.h

typedef unsigned long int 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;
extern int iswblank(wint_t);
extern wint_t towlower(wint_t);
extern wint_t towupper(wint_t);
extern wchar_t towtrans(const char *);
extern int iswalnum(wint_t);
extern int iswalpha(wint_t);
extern int iswcntrl(wint_t);
extern int iswctype(wint_t, wctype_t);
extern int iswdigit(wint_t);
extern int iswgraph(wint_t);
extern int iswlower(wint_t);
extern int iswprint(wint_t);
extern int ispunct(wint_t);
extern int iswspace(wint_t);
extern int iswupper(wint_t);
extern int iswxdigit(wint_t);
extern wctype_t wctype(const char *);
extern wint_t towctrans(wint_t, wctrans_t);

13.4.85 wordexp.h
13.5 Interface Definitions for libc

The interfaces defined on the following pages 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 in Section 13.3 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
 Tổ_C_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
 Tổ_C_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
 Tổ_C_puts – alias for puts

Synopsis
 int _IO_puts(const char * __s);

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 ** __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 of 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.

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

Return Value

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

__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

As described in the Itanium™ C++ ABI, __cxa_atexit() registers a destructor
function to be called by exit() or when a shared library is unloaded. When a
shared library is unloaded, any destructor function associated with that shared
library, identified by dso_handle, shall be called with the single argument arg,
and then that function shall be removed, or marked as complete, from the list of
functions to run at exit(). On a call to exit(), any remaining functions
registered shall be called with the single argument arg. Destructor functions
shall always be called in the reverse order to their registration (i.e. the most
recently registered function shall be called first).

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.

Note: atexit() is not in the binary standard; it is only in the source standard.
__cxa_finalize

Name

__cxa_finalize — call destructors of global (or local static) C++ objects and exit functions registered with atexit

Synopsis

void __cxa_finalize(void * d);

Description

As described in the Itanium® C++ ABI, the C runtime library shall maintain a list of termination function entries containing the following information:

- A pointer to a termination function.
- An operand to be passed to the function.
- A handle identifying the home shared library of the entry.

The list is populated by entries of two kinds:

- Destructors of global (or local static) C++ objects that require destruction on exit.
- Functions registered by the user with atexit().

In the former case an entry consists of a pointer to the destructor, a pointer to the corresponding object and a handle for the home shared library of the object. In the latter case the pointer to the function is the pointer passed to atexit(), while the other pointers are NULL.

When __cxa_finalize(d) is called, it shall walk the termination function list, calling each in turn if d matches the handle of the termination function entry. If d is NULL, it shall call all the termination functions. Multiple calls to __cxa_finalize shall not result in calling termination function entries multiple times; the implementation may either remove entries or mark them finished. The termination functions shall always be called in the reverse order of their registration (i.e. the most recently registered function shall be called first).

An application shall not call __cxa_finalize() directly. The implementation shall arrange for __cxa_finalize() to be called during early shared library unload (e.g. dlclose()) with a handle to the shared library. When the main program calls exit, the implementation shall cause any remaining __cxa_atexit-registered functions to be called, either by calling __cxa_finalize(NULL), or by walking the registration list itself.

__cxa_finalize() is not in the source standard; it is only in the binary 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
The __errno_location() function shall return the address of the errno variable for the current thread.
__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
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 ISO POSIX (2003).

__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 ISO POSIX (2003), 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 ISO POSIX (2003) 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 ISO POSIX (2003),
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 ISO POSIX (2003), 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 ISO POSIX (2003), 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 ISO POSIX (2003), 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 * * _argv0, void (*init) (void), void (*fini) (void), void (*rtld_fini) (void), void (* stack_end));
```

Description

The __libc_start_main() function shall perform any necessary initialization of the execution environment, call the main function with appropriate arguments, and handle the return from main(). If the main() function returns, the return value shall be passed to the exit() function.

**Note:** While this specification is intended to be implementation independent, process and library initialization may include:

- performing any necessary security checks if the effective user ID is not the same as the real user ID.
- initialize the threading subsystem.
- registering the rtld_fini to release resources when this dynamic shared object exits (or is unloaded).
- registering the fini handler to run at program exit.
- calling the initializer function (*init()).
- calling main() with appropriate arguments.
- calling exit() with the return value from main().

This list is an example only.

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

See Also

The section on Process Initialization in each of the architecture specific parts of ISO/IEC 23360.

__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>
void * __mempcpy(void * restrict dest, const void * restrict src,
size_t n);

Description
__mempcpy() copies n bytes of src to dest, returning a pointer to the byte 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>
void * __rawmemchr(const void *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), void (*parent) (void),
void (*child) (void), 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 — alias for stpcpy

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

Description
The __stpcpy() function has the same specification as the 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
strtol(__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) shall behave as wcstod(nptr, endptr)
as specified by ISO POSIX (2003).
__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) shall behave as wcstof(nptr, endptr)
as specified in ISO POSIX (2003).
__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) shall behave as wcstol(nptr,
endptr, base) as specified by ISO POSIX (2003).
__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**

```c
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) shall behave as wcstold(nptr, endptr) as specified by ISO POSIX (2003).

__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**

```c
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)() shall behave as wcstoul(nptr, endptr, base)() as specified by ISO POSIX (2003).

__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() function shall implement the mknod() interface from ISO POSIX (2003).

The value of ver shall be 1 or the behavior of __xmknod() is undefined.

__xmknod(1, path, mode, dev) shall behave as mknod(path, mode, dev) as

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

Note: The mknod() function is not in the binary standard; it is only in the source
standard.

__xpg_basename

Name
__xpg_basename — return the last component of a file name

Synopsis
#include <libgen.h>
char * __xpg_basename(const char *path);

Description
The __xpg_basename() function shall return a pointer to the final component of
the pathname named by path, as described in ISO POSIX (2003) basename().

This function is not in the source standard, it is only in the binary standard.

Return Value
__xpg_sigpause

Name

__xpg_sigpause — remove a signal from the signal mask and suspend the thread

Synopsis

#include <signal.h>
int __xpg_sigpause(int sig);

Description

The __xpg_sigpause() function shall implement the sigpause() described in ISO POSIX (2003).
This function is not in the source standard, it is only in the binary standard.

Return Value


__xpg_strerror_r

Name

__xpg_strerror_r — return string describing error number

Synopsis

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

Description

The __xpg_strerror_r() function shall map the error number in errnum to a locale-dependent error message string and shall return the string in the buffer pointed to by strerrorbuf, with length buflen, as described in ISO POSIX (2003) strerror_r().
This function is not in the source standard, it is only in the binary standard.

Return Value


__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 implement stat(path, stat_buf) as

__lxstat(3, path, stat_buf) shall implement lstat(path, stat_buf) as

__fxstat(3, fildes, stat_buf) shall implement fstat(fildes, stat_buf) as

__xstat64(), __lxstat64(), and __fxstat64() 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**

The global variable _nl_msg_cat_cntr is incremented each time a new catalog is loaded. This variable is only in the binary standard; it is not in the source standard.

_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

```c
#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 retracting 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.

Appropriate privilege is required to adjust the system time.

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 does not have appropriate privilege.
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 printf(), 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().

basename

Name
basename — return the last component of a file name

Synopsis
#include <libgen.h>
char * basename(const char * path);

Description
In the source standard, basename() is implemented as a macro causing it to behave as described in ISO POSIX (2003), and is equivalent to the function __xpg_basename(). If the macro is undefined, basename() from the binary standard is used, with differences as described here:
The string identified by path shall not be modified.
If path is "/", or ends with a trailing ‘/’ character, the basename() function shall return a pointer to an empty string.

Return Value
On success, the basename() function shall return a pointer to the final component of path. Otherwise, it shall return a null pointer.

See Also
__xpg_basename()
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 for 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 overrides 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.

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, dgettext, ngettext, dgettext, 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.

Note: Applications that wish to use chdir() should always use absolute pathnames to avoid inadvertently 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
ggettext, dgettext, ngettext, dgettext, 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**

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

**Description**
The `cfsetspeed()` function shall set the input and output speeds in `t` to the value specified by `speed`. 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.

**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

**daemon**

**Name**
daemon — run in the background

**Synopsis**

```c
#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, 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 `dcngettext()`. 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`, `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().

The dgettext() function shall search the currently selected message catalogs in 
the domain domainname for a string identified by the string msgid. If a string is 
located, that string shall be returned. The domain specified by domainname 
applies to the currently active LC_MESSAGE locale. The default domain shall not 
be changed by a call to dgettext().

Note: The usage of domainname 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.

The dgettext() function is equivalent to dcgettext(domainname, msgid, 
LC_MESSAGES).

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() shall not modify the errno global variable.

See Also

ggettext, dgettext, ngettext, dggettext, dcgettext, dcgettext, textdomain, 
bindtextdomain, bind_textdomain_codeset
### dngettext

#### Name

dngettext — perform lookup in message catalog for the current locale

#### Synopsis

```c
#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 dcngettext() for more information.

#### See Also

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

### duplocale

#### Name

duplocale — provide new handle for selection of locale

#### Synopsis

```c
#include <locale.h>
locale_t duplocale(locale_t locale);
```

#### Description

The duplocale() function shall provide a new locale object based on the locale object provided in `locale`, suitable for use in the newlocale() or uselocale() functions. The new object may be released by calling freelocale().

#### Return Value

On success, the duplocale() function shall return a locale object. Otherwise, it shall return NULL, and set errno to indicate the error.

#### Errors

The duplocale() function shall fail if:

ENOMEM

Insufficient memory.

#### See Also

setlocale(), freelocale(), newlocale(), uselocale()
endutent

Name
endutent — access utmp file entries

Synopsis
#include <utmp.h>
void endutent(void);

Description
endutent() closes the utmp file. It should be called when the user code is done accessing the file with the other functions.

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
#include <err.h>
#include <err.h>
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_programe(), 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 ISO POSIX (2003), only an application sets fcntl() flags, for example O_LARGEFILE. However, this specification also allows an implementation to set the O_LARGEFILE flag in the case where the programming environment is one of _POSIX_V6_ILP32_OFFBIG, _POSIX_V6_LP64_OFF64, _POSIX_V6_LPBIG_OFFBIG. See getconf and e99 in ISO POSIX (2003) for a description of these environments. Thus, calling fcntl() with the F_GETFL command may return O_LARGEFILE as well as flags explicitly set by the application in the case that both the implementation and the application support an off_t of at least 64 bits.
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**

```c
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 oring) 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.

- **EBADF**
  - `fd` is not a not an open file descriptor.

- **EINTR**
  - While waiting to acquire a lock, the call was interrupted by delivery of a signal caught by a handler.

- **EINVAL**
  - The operation is invalid.

- **EWOULDBLOCK**
  - The implementation ran out of memory for allocating lock records.
freelocale

Name
freelocale — free a locale object

Synopsis
#include <locale.h>
void freelocale(locale_t locale);

Description
The freelocale() function shall free the locale object locale, and release any resources associated with it.

Return Value
None.

Errors
None defined.

See Also
setlocale(), newlocale(), duplocale(), uselocale()

fscanf

Name
fscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences
The %s, %S and %l conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.
fstatfs

Name
fstatfs — (deprecated)

Synopsis
#include <sys/statfs.h>
int fstatfs(int fd, struct statfs *buf);

Description
The fstatfs() function returns information about a mounted file system. The file system is identified by fd, a file descriptor of an open file within the mounted filesystem. The results are placed in the structure pointed to by buf. Fields that are undefined for a particular file system shall be set to 0.

Note: Application developers should use the fstatvfs() function to obtain general file system information. Applications should only use the fstatfs() function if they must determine the file system type, which need not be provided by fstatvfs().

Return Value
On success, the fstatfs() function shall return 0 and set the fields of the structure identified by buf accordingly. On error, the fstatfs() function shall return -1 and set errno accordingly.

Errors
EBADF
  fd is not a valid open file descriptor.

EFAULT
  buf points to an invalid address.

EIO
  An I/O error occurred while reading from or writing to the file system.

ENOSYS
  The filesystem fd is open on does not support statfs().
fstatfs64

Name
fstatfs64 — (deprecated)

Synopsis
#include <sys/statfs.h>
int fstatfs64(int fd, struct statfs64 *buf);

Description
The fstatfs64() function returns information about a mounted file system.
The file system is identified by fd, a file descriptor of an open file within the
mounted filesystem. The results are placed in the structure pointed to by buf.
Fields that are undefined for a particular file system shall be set to 0.
fstatfs64() is the 64-bit version of fstatfs().

Note: Application developers should use the fstatvfs64() function to obtain
general file system information. Applications should only use the fstatfs64() function
if they must determine the file system type, which need not be provided by
fstatvfs64().

Return Value
On success, the fstatfs64() function shall return 0 and set the fields of the
structure identified by buf accordingly. On error, the fstatfs64() function
shall return -1 and set errno accordingly.

Errors
See fstatfs().

fwscanf

Name
fwscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003),
except as noted below.

Differences
The %s, %S and %[ conversion specifiers shall accept an option length modifier a,
which shall cause a memory buffer to be allocated to hold the string converted.
In such a case, the argument corresponding to the conversion specifier should
be a reference to a pointer value that will receive a pointer to the allocated
buffer. If there is insufficient memory to allocate a buffer, the function may set
errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion
specifier for hexadecimal float values. While this conversion specifier should be
supported, a format specifier such as "%aseconds" will have a different meaning on
an LSB conforming system.
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 than the length of
the string to be copied, getdomainname() shall either truncate the string to
namelen characters and place it in name (without a terminating null character),
or shall fail with EINVAL.

Note: The NIS domain name is not the same as the domain portion of a fully
qualified domain name (for example, in DNS).
The LSB does not include other NIS functions, nor does it specify how NIS may
affect other database functions. No conforming application can make use of this
information beyond noting whether or not the domain name has been set. If the
name is set to a value other than the string "(none)", the application should not
imply that NIS is in use. Similarly, if it is set to "(none)", the application should not
assume that NIS is not in use, although NIS functionality may be restricted in this
case.

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

Errors
EINVAL
name is 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 remove this interface. Application developers should avoid
using this interface where possible.
getdtablesiz

Name

getdtablesiz — get file descriptor table size (DEPRECATED)

Synopsis

#include <unistd.h>
int getdtablesiz (void);

Description

The function getdtablesiz() returns the number of files a process can have open.

Note: The getdtablesiz() function is deprecated. Portable applications should call getrlimit() with the RLIMIT_NOFILE option instead.

Return Value

The getdtablesiz() function returns the current soft limit as if obtained by a call to getrlimit() with the RLIMIT_NOFILE option.

Errors

No errors are defined.
getgrouplist

Name
getgrouplist — get groups a user belongs to

Synopsis
#include <grp.h>
int getgrouplist(const char * user, gid_t group, gid_t * groups, int * ngroups);

Description
The getgrouplist() function shall fill in the array groups with the supplementary groups for the user specified by user. On entry, ngroups shall refer to an integer containing the maximum number of gid_t members in the groups array. The group group shall also be included.

Return Value
If on entry the value referenced by ngroups was greater than or equal to the number of supplementary group identifiers to be copied to the array identified by groups, getgrouplist() shall return the number of group identifiers actually copied, and shall set the value referenced by ngroups to this value.

If on entry the value referenced by ngroups was less than to the number of supplementary group identifiers, getgrouplist() shall return -1. The initial *ngroups entries in groups may be overwritten.

If user does not refer to a valid user on the system, getgrouplist() shall return 1, and set the value referenced by ngroups to 1, reflecting the group supplied as group.

Errors
None defined.

See Also
getgroups()
gethostbyaddr_r

Name

gethostbyaddr_r — find network host database entry matching host name
(DEPRECATED)

Synopsis

#include <netdb.h>
int gethostbyaddr_r(const void * restrict addr, socklen_t len, int type, struct hostent * restrict result_buf, char * restrict buf, size_t buflen, struct hostent ** restrict result, int * h_errnop);

Description

Note: The gethostbyaddr_r() function is deprecated; applications should use getaddrinfo() instead.

gethostbyaddr_r() is a reentrant version of gethostbyaddr() that searches the network host database for a host address match.

The gethostbyaddr_r() function shall search the network host database for an entry of address family type with the host with address addr. The len argument contains the length of the address referenced by addr.

If type is AF_INET, the addr argument shall be an in_addr structure. If type is AF_INET6, the addr argument shall be an in6_addr structure. If type is any other value, the behavior is unspecified.

The application must provide a buffer for the gethostbyaddr_r() to use during the lookup process. The buffer is referenced by buf, and is of size buflen. If the buffer is not of sufficient size, gethostbyaddr_r() may fail and return ERANGE. If a matching entry is found in the database, gethostbyaddr_r() shall copy the relevant information to the application supplied hostent structure referenced by result_buf, and return a pointer to this structure in *result. If no matching entry is found, *result shall be set to a null pointer. Additional error information shall be set in the variable referenced by h_errnop.

Return Value

On success, the gethostbyaddr_r() function shall return zero. If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small. If the gethostbyaddr_r() function returns returns any other value, then the variable referenced by h_errnop shall be set to indicate the cause as for gethostbyaddr().
gethostbyname2

Name

gethostbyname2 — find network host database entry matching host name
(DEPRECATED)

Synopsis

int gethostbyname2(const char * restrict name, int af);

Description

Note: The gethostbyname2() function is deprecated; applications should use
getaddrinfo() instead.

The gethostbyname2() function shall search the network host database for an
entry with name name. This function is similar to the gethostbyname() function
but additionally allows the search to be restricted to a particular address family
specified by af.

Return Value

On success, the gethostbyname_r() function shall return zero. If the return
value was ERANGE, the size of the buffer buf, indicated by buflen, was too
small. If the gethostbyname_r() function returns returns any other value, then
the variable referenced by h_errno shall be set to indicate the cause as for
call to gethostbyname().
gethostbyname2_r

Name

gethostbyname2_r — find network host database entry matching host name
(DEPRECATED)

Synopsis

int gethostbyname2_r(const char * restrict name, int af, struct hostent * restrict result_buf, char * restrict buf, size_t buflen, struct hostent ** restrict result, int * restrict h_errno);

Description

Note: The gethostbyname2_r() function is deprecated; applications should use getaddinfo() instead.

The gethostbyname2_r() function shall search the network host database for an entry with name name. gethostbyname2_r() is a reentrant version of gethostbyname2(). These functions are similar to the gethostbyname() and gethostbyname_r() functions but additionally allow the search to be restricted to a particular address family specified by af.

The application must provide a buffer for the gethostbyname2_r() function to use during the lookup process. The buffer is referenced by buf, and is of size buflen. If the buffer is not of sufficient size, gethostbyname_r() may fail and return ERANGE. If a matching entry is found in the database, gethostbyname_r() shall copy the relevant information to the application-supplied hostent structure referenced by result_buf, and return a pointer to this structure in *result. If no matching entry is found,*result shall be set to a null pointer. Additional error information shall be set in the variable referenced by h_errno.

Return Value

On success, the gethostbyname2_r() function shall return zero. If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small. If the gethostbyname2_r() function returns returns any other value, then the variable referenced by h_errno shall be set to indicate the cause as for gethostbyname_r().
gethostbyname_r

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

Synopsis

```c
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_errno);
```

Description

Note: The `gethostbyname_r()` function is deprecated; applications should use `getaddrinfo()` instead.

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

The `gethostbyname_r()` function shall search the network host database for an entry with name `name`.

The application must provide a buffer for the `gethostbyname_r()` to use during the lookup process. The buffer is referenced by `buf`, and is of size `buflen`. If the buffer is not of sufficient size, `gethostbyname_r()` may fail and return `ERANGE`. If a matching entry is found in the database, `gethostbyname_r()` shall copy the relevant information to the application supplied `hostent` structure referenced by `result_buf`, and return a pointer to this structure in `*result`. If no matching entry is found, `*result` shall be set to a null pointer. Additional error information shall be set in the variable referenced by `h_errno`.

Return Value

On success, the `gethostbyname_r()` function shall return zero. If the return value was `ERANGE`, the size of the buffer `buf`, indicated by `buflen`, was too small. If the `gethostbyname_r()` function returns any other value, then the variable referenced by `h_errno` shall be set to indicate the cause as for `gethostbyname()`.

getloadavg

Name
getloadavg — get system load averages

Synopsis

```c
#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

g 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.

Note: 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`.

Note: 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 '}'.

© 2007 Linux Foundation
• '?' 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

g 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

g 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.

If longindex is not NULL, it points to a variable which is set to the index of the
long option relative to longopts.

Return Value

gopt_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.

gopt_long_only

Name

gopt_long_only — parse command line options

Synopsis

#define _GNU_SOURCE

© 2007 Linux Foundation
#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.

Note: The getopt_long_only() function is intended only for supporting certain programs whose command line syntax was designed before the Utility Syntax Guidelines of ISO POSIX (2003) were developed. New programs should generally call getopt_long() instead, which provides the --option syntax for long options, which is preferred by GNU and consistent with ISO POSIX (2003).

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.

getpagesize

Name

getpagesize — get memory page size (DEPRECATED)

Synopsis

#include <unistd.h>
int getpagesize (void);

Description

The function getpagesize() returns the number of bytes in a memory page.

Note: The getpagesize() function is deprecated. Portable applications should use sysconf(_SC_PAGE_SIZE) instead.

Return Value

The getpagesize() function returns the current page size.

Errors

No errors are defined.

getsockopt

Name

getsockopt — get socket options

Synopsis

#include <sys/socket.h>
#include <netinet/ip.h>
int getsockopt(int socket, int level, int option_name, void * restrict option_value, socklen_t * restrict option_len);

Description
The `getsockopt()` function shall behave as specified in ISO POSIX (2003), with the following extensions.

IP Protocol Level Options
If the `level` parameter is `IPPROTO_IP`, the following values shall be supported for `option_name` (see RFC 791:Internet Protocol for further details):

IP_OPTIONS
Get the Internet Protocol options sent with every packet from this socket. The `option_value` shall point to a memory buffer in which the options shall be placed; on entry `option_len` shall point to an integer value indicating the maximum size of the memory buffer, in bytes. On successful return, the value referenced by `option_len` shall be updated to the size of data copied to the buffer. For IPv4, the maximum length of options is 40 bytes.

IP_TTL
Get the current unicast Internet Protocol Time To Live value used when sending packets with this socket. The `option_value` shall point to a buffer large enough to hold the time to live value (at least 1 byte), and `option_len` shall point to an integer value holding the maximum size of that buffer. On successful return, the value referenced by `option_len` shall be updated to contain the number of bytes copied into the buffer, which shall be no larger than the initial value, and `option_value` shall point to an integer containing the time to live value.

IP_TOS
Get the Internet Protocol type of service indicator used when sending packets with this socket. The `option_value` shall point to a buffer large enough to hold the type of service indicator (at least 1 byte), and `option_len` shall point to an integer value holding the maximum size of that buffer. On successful return, the value referenced by `option_len` shall be updated to contain the number of bytes copied into the buffer, which shall be no larger than the initial value, and `option_value` shall point to an integer containing the time to live value.
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, dgettext, 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() function 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**

```c
#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().

**inet_aton**

**Name**

inet_aton — Internet address manipulation routine

**Synopsis**

```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int inet_aton(const char * cp, struct in_addr * inp);
```

**Description**

inet_aton() converts the Internet host address `cp` from the standard IPv4 numbers-and-dots notation into binary data and stores it in the structure that `inp` points to.

inet_aton() returns a nonzero value if the address is valid, 0 if not.

**Note:** Note that on some LSB architectures, the host byte order is Least Significant Byte first, whereas the network byte order, as used on the Internet, is Most Significant Byte first.

**initgroups**

**Name**

initgroups — initialize the supplementary group access list

**Synopsis**

```c
#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**

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

**Description**

The `ioctl()` function shall manipulate the underlying device parameters of special files. `fildes` 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**
  
  `fildes` is not a valid descriptor.

- **EFAULT**
  
  The third parameter references an inaccessible memory area.

- **ENOTTY**
  
  `fildes` is not associated with a character special device.

- **EINVAL**
  
  `request` or the third parameter is not valid.

**Relationship to POSIX (Informative)**

It should be noted that ISO POSIX (2003) contains an interface named `ioctl()`. The LSB only defines behavior when `fildes` refers to a socket (see `sockio`) or terminal device (see `ttyio`), while ISO POSIX (2003) only defines behavior when `fildes` refers to a STREAMS device. An implementation may support both behaviors; the LSB does not require any STREAMS support.
sockio

**Name**

sockio – socket ioctl commands

**Synopsis**

```c
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <net/if.h>
```
#include <netinet/in.h>

int ioctl(int sockfd, int request, void * argp);

**Description**

Socket `ioctl()` commands are a subset of the `ioctl()` calls, which can perform a variety of functions on sockets. `sockfd` shall be an open file descriptor referring to a socket (see the `socket()` or `accept()` functions).

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 (Deprecated)**

Get the interface configuration list for the system.

*Note:* The `SIOCGIFCONF` interface is superceded by the `if_nameindex()` family of functions (see ISO POSIX (2003)). A future version of this specification may withdraw this value for `request`.

`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` shall return a nonnegative value.

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

**SIOCGIFFLAGS**

Get 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**

Get 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.

**SIOCGIFBROADCAST**

Get the interface broadcast 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_broadcast` field is set with the interface broadcast address.

**SIOCGIFDSTADDR**
Get the point-to-point address for the given interface. \texttt{argp} shall point to a \texttt{ifreq} structure. Before calling, the caller should fill in the \texttt{ifr_name} field with the interface name, and upon return, the \texttt{ifr_dstaddr} field is set with the point-to-point address.

\textbf{SIOCGIFNAME}

Get the name of an interface. \texttt{argp} shall point to a \texttt{ifreq} structure. Before calling, the caller should fill in the \texttt{ifr_ifindex} field with the number (index) of the interface, and upon return, the \texttt{ifr_name} field is set with the interface name.

\textbf{SIOCGIFNETMASK}

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

\textbf{SIOCGIFMTU}

Get the Maximum Transmission Unit (MTU) size for the given interface. \texttt{argp} shall point to a \texttt{ifreq} structure. Before calling, the caller should fill in the \texttt{ifr_name} field with the interface name, and upon return, the \texttt{ifr_ifru.ifru_mtu} field is set with the MTU. Note: The range of valid values for MTU varies for an interface depending on the interface type.

\textbf{FIONREAD}

Get the amount of queued unread data in the receive buffer. \texttt{argp} shall point to an integer where the result is to be placed.

\textbf{Return Value}

On success, if \texttt{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 \texttt{errno} is set appropriately.

\textbf{Errors}

EBADF

\texttt{sockfd} is not a valid descriptor.

EFAULT

\texttt{argp} references an inaccessible memory area.

ENOTTY

The specified \texttt{request} does not apply to the kind of object that the descriptor \texttt{sockfd} references.

EINVAL

Either \texttt{request} or \texttt{argp} is invalid.

ENOTCONN

© 2007 Linux Foundation
The operation is only defined on a connected socket, but the socket wasn't connected.

ttyio

Name
ttyio — tty ioctl commands

Synopsis
#include <sys/ioctl.h>
#include <fcntl.h>
int ioctl(int fd, unsigned long request, int *argp);

Description
Tty ioctl commands are a subset of the ioctl() calls, which can perform a variety of functions on tty devices. fd shall be an open file descriptor referring to a terminal device.
The following ioctl()s are provided:

TIOCGWINSZ
Get the size attributes of the terminal or pseudo-terminal identified by fd. On entry, argp shall reference a winsize structure. On return, the structure will have ws_row set to the number of rows of text (i.e. lines of text) that can be viewed on the device, and ws_col set to the number of columns (i.e. text width).

Note: The number of columns stored in ws_col assumes that the terminal device is using a mono-spaced font.

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

Errors
EBADF
fd is not a valid descriptor.

EFAULT
argp references an inaccessible memory area.

EINVAL
request and argp are not valid.
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

link

Name

link — create a link to a file

Synopsis

#include <unistd.h>
int link(const char * path1, const char * path2);

Description

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

Need Not Follow Symlinks

ISO POSIX (2003) specifies that pathname resolution shall follow symbolic links during pathname resolution unless the function is required to act on the symbolic link itself, or certain arguments direct that the function act on the symbolic link itself. The link() function in ISO POSIX (2003) contains no such requirement to operate on a symbolic link. However, a conforming LSB implementation need not follow a symbolic link for the path1 argument.
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 shift 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, \((\text{size_t})(-1)\) is returned, and \( \text{errno} \) is set to EILSEQ.

- The \( \text{nms} \) limit forces a stop, or \( \text{len} \) non-L\(\backslash0\) 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 \( \backslash0 \) (which has the side effect of bringing back \( ps \) to the initial state). In this case, \( src \) is set to \( \text{NULL} \), and the number of wide characters written to \( dest \), excluding the terminating L\(\backslash0\) character, is returned.

If \( dest \) is \( \text{NULL} \), \( \text{len} \) 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 \( \text{NULL} \) pointer, a static anonymous state only known to \text{mbsnrtowcs()} is used instead.

The programmer shall ensure that there is room for at least \( \text{len} \) wide characters at \( dest \).

**Return Value**

\text{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, \((\text{size_t})(-1)\) is returned, and the global variable \( \text{errno} \) is set to EILSEQ.

**Notes**

The behavior of \text{mbsnrtowcs()} depends on the \text{LC_CTYPE} category of the current locale.

Passing \( \text{NULL} \) as \( ps \) is not multi-thread safe.

**memmem**

**Name**

\text{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 occurrence 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()
mremap

Name

mremap — remap a virtual memory address

Synopsis

#include <sys/mman.h>
void * mremap(void * old_address, size_t old_size, size_t new_size,
int flags);

Description

The mremap() function expands (or shrinks) an existing memory mapping,
potentially moving it at the same time, depending on the flags argument and
the available virtual address space.

old_address is the old address of the virtual memory block to be resized. Note
that old_address must be page aligned. old_size is the old size of the virtual
memory block. new_size is the requested size of the virtual memory block after
the resize.

In Linux the memory is divided into pages. A user process has (one or) several
linear virtual memory segments. Each virtual memory segment has one or more
mappings to real memory pages (in the page table). Each virtual memory
segment has its own protection (access rights), which may cause a segmentation
violation if the memory is accessed incorrectly (e.g., writing to a read-only
segment). Accessing virtual memory outside of the segments will also cause a
segmentation violation.

mremap() uses the Linux page table scheme. mremap() changes the mapping
between virtual addresses and memory pages. This can be used to implement a
very efficient form of realloc().

The flags bit-mask argument may be 0, or include the following flag:

MREMAP_MAYMOVE

By default, if there is not sufficient space to expand a mapping at its current
location, then mremap() fails. If this flag is specified, then the kernel is
permitted to relocate the mapping to a new virtual address, if necessary. If
the mapping is relocated, then absolute pointers into the old mapping
location become invalid (offsets relative to the starting address of the
mapping should be employed).

MREMAP_FIXED

This flag serves a similar purpose to the MAP_FIXED flag of mmap(). If this
flag is specified, then mremap() accepts a fifth argument, void
*new_address, which specifies a pagealigned address to which the
mapping must be moved. Any previous mapping at the address range
specified by new_address and new_size is unmapped. If MREMAP_FIXED is
specified, then MREMAP_MAYMOVE must also be specified.

If the memory segment specified by old_address and old_size is locked
(using mlock() or similar), then this lock is maintained when the segment is
resized and/or relocated. As a consequence, the amount of memory locked by
the process may change.

Return Value
The mremap() function returns a pointer to the new virtual memory area on success. On error, the value MAP_FAILED is returned, and errno is set appropriately.

Errors

EAGAIN
The caller tried to expand a memory segment that is locked, but this was not possible without exceeding the RLIMIT_MEMLOCK resource limit.

EFAULT
"Segmentation fault." Some address in the range old_address to old_address+old_size is an invalid virtual memory address for this process. You can also get EFAULT even if there exist mappings that cover the whole address space requested, but those mappings are of different types.

EINVAL
An invalid argument was given. Possible causes are: old_address was not page aligned; a value other than MREMAP_MAYMOVE or MREMAP_FIXED was specified in flags; new_size was zero; new_size or new_address was invalid; or the new address range specified by new_address and new_size overlapped the old address range specified by old_address and old_size; or MREMAP_FIXED was specified without also specifying MREMAP_MAYMOVE.

ENOMEM
The memory area cannot be expanded at the current virtual address, and the MREMAP_MAYMOVE flag is not set in flags, or, there is not enough (virtual) memory available.
newlocale

Name
newlocale — allocate a locale object

Synopsis
#include <locale.h>
locale_t newlocale(int category_mask, const char *locale, locale_t base);

Description
The newlocale() function shall initialize a locale object. If base is NULL, then
ewlocale() shall first allocate the object; otherwise it shall use the locale object
referenced by base.

The object shall be initialized for the locale named by locale, and for the
categories selected in category_mask. The category_mask value is a bitwise
inclusive OR of the required LC_name_MASK values, or the value LC_ALL_MASK.

Return Value
On success, the newlocale() function shall return the initialized locale object.
Otherwise, it shall return NULL, and set errno to indicate the error.

Errors
The newlocale() function shall fail if:

ENOMEM
Insufficient memory.

EINVAL
An invalid category_mask was provided, or the locale was NULL.

ENOENT
For any of the categories in category_mask, the locale data is not available.

Application Usage (Informative)
The only portable way to allocate a locale object is to call newlocale() with a
NULL base. The allocated object may be reinitialized to a new locale by passing
it back to newlocale(). The new object may be released by calling
freelocale().

See Also
setlocale(), freelocale(), duplocale(), uselocale()
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 (dependent 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 message 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, dgettext, ngettext, dngettext, dcgettext, dcngettext, textdomain, bindtextdomain, bind_textdomain_codeset
pmap_getport

**Name**

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

**Synopsis**

```c
#include <rpc/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.

**Note:** 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**

```c
#include <rpc/pmap_clnt.h>

bool_t 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 non-zero if it succeeds, 0 otherwise.
pmap_unset

Name
pmap_unset — destroys RPC Binding

Synopsis
#include <rpc/pmap_clnt.h>

bool_t 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 \([\text{prognum}, \text{versnum}, *]\) and ports on the machine's RPC Bind service.

Return Value
pmap_unset() returns non-zero if it succeeds, zero otherwise.

posix_fadvise64

Name
posix_fadvise64 — File advisory information (Large File Support)

Synopsis
#include <fcntl.h>

int posix_fadvise64(int fd, off64_t offset, off64_t len, int advice);

Description
The posix_fadvise64() function is a large-file version of the posix_fadvise() function defined in ISO POSIX (2003). It shall advise the implementation on the expected behavior of the application with respect to the data in the file associated with the open file descriptor, \(fd\), starting at offset and continuing for len bytes. The specified range need not currently exist in the file. If len is zero, all data following offset is specified. The implementation may use this information to optimize handling of the specified data. The posix_fadvise() function shall have no effect on the semantics of other operations on the specified data, although it may affect the performance of other operations.

The advice to be applied to the data is specified by the advice parameter, as specified in posix_fadvise().

Return Value
On success, posix_fadvise64() shall return 0. Otherwise an error number shall be returned to indicate the error. See posix_fadvise() for possible error values.
posix_fallocate64

Name
posix_fallocate64 — file space control (Large File Support)

Synopsis
#include <fcntl.h>
int posix_fallocate64(int fd, off64_t offset, off64_t len);

Description
The posix_fallocate64() function is a large file version of
posix_fallocate(). It shall behave as posix_fallocate() in ISO POSIX
(2003), except that the offset and len arguments are off64_t objects rather
than off_t.

Return Value
See posix_fallocate().

Errors
See posix_fallocate().

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.
readdir64_r

Name

readdir64_r — read a directory (Large File Support)

Synopsis

#include <dirent.h>
int readdir64_r(DIR * dirp, struct dirent64 * entry, struct
dirent64 ** result);

Description

The readdir64_r() function is a large file version of readdir_r(). It shall
behave as readdir_r() in ISO POSIX (2003), except that the entry and result
arguments are dirent64 structures rather than dirent.

Return Value

See readdir_r().

Errors

See readdir_r().

regexec

Name

regexec — regular expression matching

Description

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

Differences

Certain aspects of regular expression matching are optional; see Regular
Expressions.
**scanf**

**Name**

scanf — convert formatted input

**Description**

The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

**Differences**

The %s, %S and %[] conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

*Note:* This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

**sched_setscheduler**

**Name**

sched_setscheduler — set scheduling policy and parameters

**Synopsis**

#include <sched.h>

int sched_setscheduler(pid_t pid, int policy, const struct sched_param *param);

**Description**

The sched_setscheduler() shall behave as described in ISO POSIX (2003), except as noted below.

**Return Value**

On success, 0 is returned instead of the former scheduling policy.
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.

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.
sethostname

Name
sethostname — set host name

Synopsis
#include <unistd.h>
#include <sys/param.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 machine. 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 indicates 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).

© 2007 Linux Foundation
setsockopt

Name
setsockopt — set socket options

Synopsis
#include <sys/socket.h>
#include <netinet/ip.h>
int setsockopt(int socket, int level, int option_name, const void *option_value, socklen_t option_len);

Description
The setsockopt() function shall behave as specified in ISO POSIX (2003), with the following extensions.

IP Protocol Level Options
If the level parameter is IPPROTO_IP, the following values shall be supported for option_name (see RFC 791:Internet Protocol for further details):

IP_OPTIONS
Set the Internet Protocol options sent with every packet from this socket. The option_value shall point to a memory buffer containing the options and option_len shall contain the size in bytes of that buffer. For IPv4, the maximum length of options is 40 bytes.

IP_TOS
Set the Type of Service flags to use when sending packets with this socket. The option_value shall point to a value containing the type of service value. The least significant two bits of the value shall contain the new Type of Service indicator. Use of other bits in the value is unspecified. The option_len parameter shall hold the size, in bytes, of the buffer referred to by option_value.

IP_TTL
Set the current unicast Internet Protocol Time To Live value used when sending packets with this socket. The option_value shall point to a value containing the time to live value, which shall be between 1 and 255. The option_len parameter shall hold the size, in bytes, of the buffer referred to by option_value.

IP_MULTICAST_TTL
Sets the Time To Live value of outgoing multicast packets for this socket. optarg shall point to an integer which contains the new TTL value. If the new TTL value is -1, the implementation should use an unspecified default TTL value. If the new TTL value is out of the range of acceptable values (0-255), setsockopt() shall return -1 and set errno to indicate the error.

IP_MULTICAST_LOOP
Sets a boolean flag indicating whether multicast packets originating locally should be looped back to the local sockets. optarg shall point to an integer which contains the new flag value.

IP_ADD_MEMBERSHIP
Join a multicast group. optarg shall point 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.
Leave a multicast group. `optval` shall point 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` shall point 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.

**Errors**

As defined in ISO POSIX (2003).

### setutent

**Name**

`setutent` — access user accounting database entries

**Synopsis**

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

**Description**

The `setutent()` function shall reset the user accounting database such that the next call to `getutent()` shall 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**

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

**Description**

The `sigandset()` function 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`.

Applications shall call `sigemptyset()` or `sigfillset()` at least once for each object of type `sigset_t` to initialize it. If an uninitialized or NULL object is passed to `sigandset()`, the results are undefined.

**Return Value**

`sigandset()` returns 0. There are no defined error returns.

**See Also**

`sigorset()`

sigisemptyset

**Name**

`sigisemptyset` – check for empty signal set

**Synopsis**

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

**Description**

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

Applications shall call `sigemptyset()` or `sigfillset()` at least once for each object of type `sigset_t` to initialize it. If an uninitialized or NULL object is passed to `sigisemptyset()`, the results are undefined.

**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. There are no defined error returns.
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() function 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.

Applications shall call sigemptyset() or sigfillset() at least once for each object of type sigset_t to initialize it. If an uninitialized or NULL object is passed to sigorset(), the results are undefined.

Return Value

sigorset() returns 0. There are no defined error returns.

See Also

sigandset()

sigpause

Name

sigpause — remove a signal from the signal mask and suspend the thread (deprecated)

Synopsis

#include <signal.h>

int sigpause(int sig);

Description

The sigpause() function is deprecated from the LSB and is expected to disappear from a future version of the LSB. Conforming applications should use sigsuspend() instead.

In the source standard, sigpause() is implemented as a macro causing it to behave as described in ISO POSIX (2003), and is equivalent to the function __xpg_sigpause(). If the macro is undefined, sigpause() from the binary standard is used, with differences as described here:

The sigpause() function shall block those signals indicated by sig and suspend execution of the thread until a signal is delivered. When a signal is delivered, the original signal mask shall be restored.

See Also

__xpg_sigpause()
sigreturn

Name

sigreturn — return from signal handler and cleanup stack frame

Synopsis

int sigreturn(struct sigcontext * scp);

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.

sscanf

Name

sscanf — convert formatted input

Description

The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

The %s, %S and % conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.
statfs

Name
statfs — (deprecated)

Synopsis
#include <sys/statfs.h>
int statfs(const char *path, (struct statfs * buf));

Description
The statfs() function returns information about a mounted file system. The
file system is identified by path, a path name of a file within the mounted
filesystem. The results are placed in the structure pointed to by
Fields that are undefined for a particular file system shall be set to 0.

Note: Application developers should use the statvfs() function to obtain general
file system information. Applications should only use the statfs() function if they
must determine the file system type, which need not be provided by statvfs().

Return Value
On success, the statfs() function shall return 0 and set the fields of the
structure identified by buf accordingly. On error, the statfs() function shall
return -1 and set errno accordingly.

Errors
ENOTDIR
A component of the path prefix of path is not a directory.

ENAMETOOLONG
path is too long.

ENOENT
The file referred to by path does not exist.

EACCES
Search permission is denied for a component of the path prefix of path.

ELOOP
Too many symbolic links were encountered in translating path.

EFAULT
buf or path points to an invalid address.

EIO
An I/O error occurred while reading from or writing to the file system.

ENOMEM
Insufficient kernel memory was available.

ENOSYS
The filesystem path is on does not support statfs().

**statfs64**

**Name**

statfs64 — (deprecated)

**Synopsis**

```c
#include <sys/statfs.h>
int statfs64(const char *path, (struct statfs64 *buf));
```

**Description**

The statfs64() function returns information about a mounted file system. The file system is identified by path, a path name of a file within the mounted filesystem. The results are placed in the structure pointed to by buf. statfs64() is the 64-bit version of statfs().

Fields that are undefined for a particular file system shall be set to 0.

**Note:** Application developers should use the statvfs64() function to obtain general file system information. Applications should only use the statfs64() function if they must determine the file system type, which need not be provided by statvfs64().

**Return Value**

On success, the statfs64() function shall return 0 and set the fields of the structure identified by buf accordingly. On error, the statfs64() function shall return -1 and set errno accordingly.

**Errors**

See fstatfs().

**stime**

**Name**

stime — set time

**Synopsis**

```c
#define _SVID_SOURCE
```
#include <time.h>
int stime(const 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 null 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 null 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);
stncpy

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 null 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 null 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() function 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` — return string describing error number

**Synopsis**

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

**Description**

In the source standard, `strerror_r()` is implemented as a macro causing it to behave as described in ISO POSIX (2003), and is equivalent to the function `__xpg_strerror_r()`. If the macro is undefined, `strerror_r()` from the binary standard is used, with differences as described here.

The `strerror_r()` function shall return a pointer to the string corresponding to `errno`. The returned pointer may point within the buffer `buf` (at most `buflen` bytes).

**Return Value**

On success, `strerror_r()` shall return a pointer to the generated message string (determined by the setting of the LC_MESSAGES category in the current locale). Otherwise, `strerror_r()` shall return the string corresponding to "Unknown error".

**See Also**

`__xpg_strerror_r()`
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
The strnlen() function shall compute the number of bytes in the array to which s points, stopping at maxlen bytes. A null byte and any bytes following it are not counted.

Return Value
The strnlen() function shall return the length of s if that is less than maxlen, or maxlen if there is no null byte in the first maxlen bytes.

Errors
No errors are defined.
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

```c
#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

```c
#define _GNU_SOURCE
```

© 2007 Linux Foundation
#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
```c
#include <sys/types.h>
#include <stdlib.h>
```
#include <limits.h>
long long strtoq(const char * nptr, char ** endptr, int base);

Description

strtoq() converts the string nptr to a quad 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()), 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.

strtoq

Name

strtoq — convert a string to an unsigned long long

Synopsis

#include <sys/types.h>
#include <stdlib.h>
#include <limits.h>
unsigned long long strtouq(const char * nptr, char ** endptr, int base);

Description

strtouq() converts the string nptr to an unsigned long long 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()), 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.

svc_register

Name

svc_register — register Remote Procedure Call interface

Synopsis

#include <rpc/rpc.h>
bool_t svc_register(SVCXPRT * xprt, rpcprog_t prognum, rpcvers_t versnum, __dispatch_fn_t dispatch, rpcprot_t 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
bool_t svc_sendreply(SVCXPRT *xprt, xdrproc_t outproc, caddr_t 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 otherwise.
svctcp_create

Name
svctcp_create — create 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() creates 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, then 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 — create a UDP-based RPC service transport

Synopsis
SVCXPRT *
svcudp_create(int sock);

Description
The svcudp_create() function shall create a UDP/IP-based RPC service transport, and return a pointer to its descriptor. The transport is associated with the socket sock, which may be RPC_ANYSOCK, in which case a new socket shall be created. If the socket is not bound to a local UDP port, then svcudp_create() shall bind it to an arbitrary port.

If svcudp_create() returns successfully, then the xp_sock field in the result shall be the transport's socket descriptor, and the xp_port field shall be the transport's port number.

Return Value
Upon successful completion, svcudp_create() shall return a pointer to a RPC service transport; otherwise, a null pointer shall be returned.
swscanf

Name
swscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences
The %s, %S and %[ conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

sysconf

Name
sysconf — Get configuration information at runtime

Synopsis
#include <unistd.h>
long sysconf(int name);

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

Extra Variables
These additional values extend the list in ISO POSIX (2003).

- _SC_PHYS_PAGES
  The number of pages of physical memory.

- _SC_AVPHYS_PAGES
  The number of currently available pages of physical memory.

- _SC_NPROCESSORS_CONF
  The number of processors configured.

- _SC_NPROCESSORS_ONLN
  The number of processors currently online (available).
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 execvp() 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 is set to indicate the error.

Errors
ENOMEM
Insufficient 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 Section 18.1, 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.
uselocale

Name
uselocale — set locale for thread

Synopsis
#include <locale.h>
locale_t uselocale(locale_t newloc);

Description
The uselocale() function shall set the locale for the calling thread to the locale specified by newloc.

If newloc is the value LC_GLOBAL_LOCALE, the thread's locale shall be set to the process current global locale, as set by setlocale(). If newloc is NULL, the thread's locale is not altered.

Return Value
The uselocale() function shall return the previous locale, or LC_GLOBAL_LOCALE if the thread local locale has not been previously set.

Errors
None defined.

See Also
setlocale(), freelocale(), duplocale(), newlocale()

utmpname

Name
utmpname — set user accounting database

Synopsis
#include <utmp.h>
int utmpname(const char * dbname);

Description
The utmpname() function shall cause the user accounting database used by the getutent(), getutent_r(), getutxent(), getutxid(), getutxline(), and pututxline() functions to be that named by dbname, instead of the system default database. See Section 16.3 for further information.

Note: The LSB does not specify the format of the user accounting database, nor the names of the file or files that may contain it.

Return Value
None.

Errors
None defined.
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() function shall behave as vfprintf(), except that vdprintf() shall write output to the file associated with the file descriptor specified by the fd argument, rather than place output on a stream (as defined by ISO POSIX (2003)).

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.

vfscanf

Name
vfscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003),
except as noted below.

Differences
The %s, %S and %c conversion specifiers shall accept an option length modifier a,
which shall cause a memory buffer to be allocated to hold the string converted.
In such a case, the argument corresponding to the conversion specifier should
be a reference to a pointer value that will receive a pointer to the allocated
buffer. If there is insufficient memory to allocate a buffer, the function may set
errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion
specifier for hexadecimal float values. While this conversion specifier should be
supported, a format specifier such as "%aseconds" will have a different meaning on
an LSB conforming system.
vfwscanf

Name
vfwscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences
The %s, %S and %[ conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

vscanf

Name
vscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences
The %s, %S and %[ conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.
vsscanf

Name

vsscanf — convert formatted input

Description

The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences

The %s, %S and %[ conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

vswscanf

Name

vswscanf — convert formatted input

Description

The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences

The %s, %S and %[ conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

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.

vscanf

Name
vscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences
The %s, %S and %l conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

wait4

Name
wait4 — wait for process termination, BSD style

Synopsis
#include <sys/types.h>
#include <sys/resource.h>


```c
#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:

- **Note**: 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.

- `EXITSTATUS(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() 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 XSI optional functionality of WCONTINUED() or WIFCONTINUED().
**warn**

**Name**

warn — formatted error messages  

**Synopsis**

```c
#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**

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

**Description**

The `warnx()` 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 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 strcpy(). It copies the wide
character string src, including the terminating null wide character code, 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 null wide character code.

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
null wide character code, 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 null wide character codes. If
the length wcslen(src) is greater than or equal to n, the string dest will not
be terminated with a null wide character code.

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**

```c
#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**

The wcscasecmp() function shall return 0 if the wide-character strings `s1` and `s2` are equal except for case distinctions. It shall return a positive integer if `s1` is greater than `s2`, ignoring case. It shall return a negative integer if `s1` is less 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**

```c
#include <wchar.h>
wchar_t * wcsdup(const wchar_t * s);
```

**Description**

The wcsdup() function is the wide-character equivalent of strdup(). The wcsdup() function shall return a pointer to a new wide character string, which is a duplicate of the wide character string pointed to by `s`. The returned pointer can be passed to free(). A null pointer is returned if the new string cannot be created.

**Return Value**

The wcsdup() function returns a pointer to a new wide-character string on success. Otherwise, it shall return NULL and set errno to indicate the error.

**Errors**

- EINVAL
  - Insufficient memory available.
**wcsncasecmp**

**Name**

wcsncasecmp — compare two fixed-size wide-character strings, ignoring case

**Synopsis**

```c
#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**

wcsncasecmp() 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**

```c
#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 null wide character code, 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 null wide character code among the first maxlen wide characters pointed to by s.
**wcsnrtombs**

**Name**

wcsnrtombs — convert a wide character string to a multi-byte string

**Synopsis**

```c
#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 shift state ps is updated.

The conversion is effectively performed by repeatedly calling:

```c
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 null wide character code, 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 null wide character code (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 null wide character code, 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 null wide character code. 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**

```c
#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**

```c
#include <wchar.h>
unsigned long long 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()`.
wscanf

Name
wscanf — convert formatted input

Description
The scanf() family of functions shall behave as described in ISO POSIX (2003), except as noted below.

Differences
The %s, %S and %[ conversion specifiers shall accept an option length modifier a, which shall cause a memory buffer to be allocated to hold the string converted. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer value that will receive a pointer to the allocated buffer. If there is insufficient memory to allocate a buffer, the function may set errno to ENOMEM and a conversion error results.

Note: This directly conflicts with the ISO C (1999) usage of %a as a conversion specifier for hexadecimal float values. While this conversion specifier should be supported, a format specifier such as "%aseconds" will have a different meaning on an LSB conforming system.

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.
xdrstdio_create

Name
xdrstdio_create — library routines for external data representation

Synopsis
#include <rpc/xdr.h>
void xdrstdio_create(XDR * xdrs, FILE * file, enum xdr_op op);

Description
The xdrstdio_create() function shall initialize the XDR stream object referred to by xdrs. The XDR stream data shall be written to, or read from, the standard I/O stream associated with file. If the operation op is XDR_ENCODE, encoded data shall be written to file. If op is XDR_DECODE, encoded data shall be read from file. If op is XDR_FREE, the XDR stream object may be used to deallocate storage allocated by a previous XDR_DECODE.

The associated destroy function shall flush the file I/O stream, but not close it.

Return Value
None.

13.6 Interfaces for libm

Table 13-33 defines the library name and shared object name for the libm library

Table 13-33 libm Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libm</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>See archLSB.</td>
</tr>
</tbody>
</table>

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

-[LSB] This Specification
-[SVID.3] SVID Issue 3

13.6.1 Math

13.6.1.1 Interfaces for Math

An LSB conforming implementation shall provide the generic functions for Math specified in Table 13-34, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-34 libm - Math Function Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>acosl</td>
<td>asin</td>
<td>asinf</td>
<td>asinh</td>
</tr>
<tr>
<td>asinhf</td>
<td>asinhl</td>
<td>asinl</td>
<td>atanh</td>
</tr>
<tr>
<td>atan2</td>
<td>atan2f</td>
<td>atan2l</td>
<td>atanhf</td>
</tr>
<tr>
<td>atanh</td>
<td>atanhl</td>
<td>atanl</td>
<td></td>
</tr>
<tr>
<td>cabs</td>
<td>cabsf</td>
<td>cabsl</td>
<td>cacos</td>
</tr>
<tr>
<td>cacosh</td>
<td>cacoshf</td>
<td>cacoshl</td>
<td>cacosf</td>
</tr>
<tr>
<td>cacos</td>
<td>carg</td>
<td>cargf</td>
<td>cargl</td>
</tr>
<tr>
<td>casin</td>
<td>casinf</td>
<td>casinh</td>
<td>casinhf</td>
</tr>
<tr>
<td>casinhl</td>
<td>casinl</td>
<td>catan</td>
<td>catanf</td>
</tr>
<tr>
<td>catanh</td>
<td>catanhf</td>
<td>catanhl</td>
<td>catanhl</td>
</tr>
<tr>
<td>cbrt</td>
<td>cbrtf</td>
<td>cbrtl</td>
<td>cbrt</td>
</tr>
<tr>
<td>ccos</td>
<td>ccosf</td>
<td>ccosh</td>
<td>ccoshf</td>
</tr>
<tr>
<td>ccosl</td>
<td>ceil</td>
<td>ceilf</td>
<td>ceill</td>
</tr>
<tr>
<td>cexp</td>
<td>cexpf</td>
<td>cexpl</td>
<td>cimag</td>
</tr>
<tr>
<td>cimagf</td>
<td>cimagl</td>
<td>clog</td>
<td>clog10</td>
</tr>
<tr>
<td>clog10</td>
<td>clog10f</td>
<td>clog1l</td>
<td></td>
</tr>
<tr>
<td>conj</td>
<td>conjf</td>
<td>conjl</td>
<td>copy</td>
</tr>
<tr>
<td>copy</td>
<td>copyf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>copysignf</td>
<td>copysignl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cos</td>
<td>cosf</td>
<td>cosl</td>
<td></td>
</tr>
<tr>
<td>cosh</td>
<td>coshf</td>
<td>coshl</td>
<td>cosl</td>
</tr>
<tr>
<td>cpow</td>
<td>cpowf</td>
<td>cpowl</td>
<td></td>
</tr>
<tr>
<td>cproj</td>
<td>cprojf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creal</td>
<td>crealf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>crealf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>creall</td>
<td>creal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creallf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csinh</td>
<td>csinhl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>csinl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csqrt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csqrtf</td>
<td>csqrtl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctanh</td>
<td>ctanhf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctanhl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drem</td>
<td>dremf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dreml</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>erf</td>
<td>erfcl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>erfcl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exp</td>
<td>expf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exp10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exp10f</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exp2</td>
<td>exp2f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>exp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expm1</td>
<td>expm1f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>expm1l</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fabs</td>
<td>fabsf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fabsl</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdim</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdimf</td>
<td>fdiml</td>
<td></td>
<td></td>
</tr>
<tr>
<td>feclearexcept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fedisableexcept</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fegetenv</td>
<td>fegetexcept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fegetexceptflag</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[LSB]</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------</td>
<td>------------------------------</td>
<td></td>
</tr>
<tr>
<td>fegetround</td>
<td>feholdexcept</td>
<td>feraiseexcept</td>
<td></td>
</tr>
<tr>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fesetexceptflag</td>
<td>fesetround</td>
<td>fetestexcept</td>
<td></td>
</tr>
<tr>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>finite [LSB]</td>
<td>finitef [LSB]</td>
<td>finitel [LSB]</td>
<td></td>
</tr>
<tr>
<td>fmal [SUSv3]</td>
<td>fmax [SUSv3]</td>
<td>fmaxf [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fmodf [SUSv3]</td>
<td>fmodl [SUSv3]</td>
<td>frexp [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>frexpl [SUSv3]</td>
<td>gamma [LSB]</td>
<td>gammal [LSB]</td>
<td></td>
</tr>
<tr>
<td>ilogbf [SUSv3]</td>
<td>ilogbl [SUSv3]</td>
<td>j0 [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>j0l [LSB]</td>
<td>j1 [SUSv3]</td>
<td>j1f [LSB]</td>
<td></td>
</tr>
<tr>
<td>jn [SUSv3]</td>
<td>jnf [LSB]</td>
<td>jnl [LSB]</td>
<td></td>
</tr>
<tr>
<td>lgammaf [SUSv3]</td>
<td>lgamma_r [LSB]</td>
<td>lgamma_l [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>llrint [SUSv3]</td>
<td>llrintf [SUSv3]</td>
<td>llrintl [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>log10f [SUSv3]</td>
<td>log10l [SUSv3]</td>
<td>log1p [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>log1pl [SUSv3]</td>
<td>log2 [SUSv3]</td>
<td>log2f [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>log [SUSv3]</td>
<td>log10 [SUSv3]</td>
<td>log1pf [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>remainderf</td>
<td>remainderl</td>
<td>remquo [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>nearbyintl</td>
<td>nextafter [SUSv3]</td>
<td>nextafterf [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>nexttoward f</td>
<td>nexttowardl [SUSv3]</td>
<td>pow [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
<td>powl [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>remquof [SUSv3]</td>
<td>remquol [SUSv3]</td>
<td>rint [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>rintf [SUSv3]</td>
<td>rintl [SUSv3]</td>
<td>round [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>scalbl [ISOC99]</td>
<td>scalbf [ISOC99]</td>
<td>scalbl [ISOC99]</td>
<td></td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for Math specified in Table 13-35, with the full mandatory functionality as described in the referenced underlying specification.

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

**Table 13-35 libm - Math Deprecated Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gammal [LSB]</td>
<td>matherr [SVID.3]</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Math specified in Table 13-36, with the full mandatory functionality as described in the referenced underlying specification.

**Table 13-36 libm - Math Data Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signgam [SUSv3]</td>
<td></td>
</tr>
</tbody>
</table>

**13.7 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.
This specification uses the ISO C (1999) 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.

### 13.7.1 complex.h

```c
#define complex _Complex

extern double cabs(double complex);
extern float cabsf(float complex);
extern long double cabsl(long double complex);
extern double complex cacos(double complex);
extern float complex cacosf(float complex);
extern long double complex cacosh(long double complex);
extern long double complex cacosl(long double complex);
extern double complex carg(double complex);
extern float cargf(float complex);
extern long double complex cargl(long double complex);
extern double complex casin(double complex);
extern float complex casinf(float complex);
extern long double complex casinl(long double complex);
extern double complex casinh(double complex);
extern float complex casinhf(float complex);
extern long double complex casinhl(long double complex);
extern long double complex casinl(long double complex);
extern double complex catan(double complex);
extern float complex catanf(float complex);
extern long double complex catanl(long double complex);
extern double complex catanh(double complex);
extern float complex catanhf(float complex);
extern long double complex catanhl(long double complex);
extern long double complex catanl(long double complex);
extern double complex ccos(double complex);
extern float complex ccosf(float complex);
extern long double complex ccosl(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexpl(long double complex);
extern double complex cimag(double complex);
extern float cimagf(float complex);
extern long double complex cimagl(long double complex);
extern double complex clog(double complex);
extern double complex clog10(double complex);
extern double complex clog10f(float complex);
extern double complex clog10l(long double complex);
extern double complex cproj(double complex);
extern float complex cprojf(float complex);
extern long double complex cprojl(float complex);
extern double complex conj(double complex);
extern float complex conjf(float complex);
extern long double complex conjl(long double complex);
extern double complex cpow(double complex, double complex);
extern float complex cpowf(float complex, float complex);
extern long double complex cpowl(long double complex, long double complex);
extern double complex creal(double complex);
extern float crealf(float complex);
```

© 2007 Linux Foundation
13 Base Libraries

ISO/IEC 23360 Part 1:2007(E)

extern long double creall(long double complex);
extern double complex csin(double complex);
extern float complex csinf(float complex);
extern double complex csinh(double complex);
extern float complex csinhf(float complex);
extern long double complex csinhl(long double complex);
extern long double complex csinl(long double complex);
extern double complex csqrt(double complex);
extern float complex csqrtf(float complex);
extern long double complex csqrtl(long double complex);
extern double complex ctan(double complex);
extern float complex ctanf(float complex);
extern double complex ctanh(double complex);
extern float complex ctanhf(float complex);
extern long double complex ctanhl(long double complex);
extern long double complex ctanl(long double complex);

13.7.2 fenv.h

extern int fedisableexcept(int);
extern int feenableexcept(int);
extern int fegetexcept(void);
extern int feclearexcept(int);
extern int fegetenv(fenv_t *);
extern int fegetexceptflag(fexcept_t *, int);
extern int fegetround(void);
extern int feholdexcept(fenv_t *);
extern int feraiseexcept(int);
extern int fesetenv(const fenv_t *);
extern int fesetexceptflag(const fexcept_t *, int);
extern int fesetround(int);
extern int fetestexcept(int);
extern int feupdateenv(const fenv_t *);

13.7.3 math.h

#define DOMAIN 1
#define SING 2

struct exception {
    int type;
    char *name;
    double arg1;
    double arg2;
    double retval;
};

#define FP_NAN 0
#define FP_INFINITE 1
#define FP_ZERO 2
#define FP_SUBNORMAL 3
#define FP_NORMAL 4

#define isnormal(x) (fpclassify (x) == FP_NORMAL)

#define HUGE_VAL 0x1.0p2047
#define HUGE_VALF 0x1.0p255f

#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

246 © 2007 Linux Foundation
#define M_PI_4  0.78539816339744830962
#define M_2_SQRTPI      1.12837916709551257390
#define M_SQRT2 1.41421356237309504880
#define M_LOG2E 1.4426950408889634074
#define M_PI_2  1.57079632679489661923
#define M_LN10  2.30258509299404568402
#define M_E     2.7182818284590452354
#define M_PI    3.14159265358979323846
#define INFINITY        HUGE_VALF

#define MATH_ERRNO      1
#define MATH_ERREXCEPT    2

#define isunordered(u, v)       \
   (__extension__((typeof)(u) __u = (u); typeof(v) \n    _v = (v); fpclassify (__u) == FP_NAN || fpclassify (__v) == FP_NAN; ))

#define islessgreater(x, y)     \
   (__extension__((typeof)(x) __x = (x); typeof(y) \n    _y = (y); !_isunordered (__x, __y) && (__x < __y || __y < __x); ))

#define isless(x,y)     \
   (__extension__((typeof)(x) __x = (x); typeof(y) \n    _y = (y); !_isunordered (__x, __y) && __x < __y; ))

#define islessequal(x, y)       \
   (__extension__((typeof)(x) __x = (x); typeof(y) \n    _y = (y); !_isunordered (__x, __y) && __x <= __y; ))

#define isgreater(x,y)  \
   (__extension__((typeof)(x) __x = (x); typeof(y) \n    _y = (y); !_isunordered (__x, __y) && __x > __y; ))

#define isgreaterequal(x,y)     \
   (__extension__((typeof)(x) __x = (x); typeof(y) \n    _y = (y); !_isunordered (__x, __y) && __x >= __y; ))

extern int __finite(double);
extern int __finitef(float);
extern int __finitel(long double);
extern int __isinf(double);
extern int __isinff(float);
extern int __isinfl(long double);
extern int __isnan(double);
extern int __isnanf(float);
extern int __isnanl(long double);
extern int __signbit(double);
extern int __signbitf(float);
extern int __fpclassify(double);
extern int __fpclassifyf(float);
extern int signgam;
extern double copysign(double, double);
extern double frexp(double, int *);
extern double ldexp(double, int);
extern double modf(double, double *);
extern double acos(double);
extern double acosh(double);
extern double asin(double);
extern double asinh(double);
extern double atanh(double);
extern double asin(double);
extern double atan(double);
extern double atan2(double, double);
extern double cbrt(double);
extern double ceil(double);
extern double cos(double);
extern double cosh(double);
extern double erf(double);
extern double erfc(double);
extern double exp(double);
extern double expm1(double);
extern double fabs(double);
extern double floor(double);
extern double fmod(double, double);
extern double gamma(double);
extern double hypot(double, double);
extern int ilogb(double);
extern double j0(double);
extern double j1(double);
extern double jn(int, double);
extern double lgamma(double);
extern double log(double);
extern double log10(double);
extern double log1p(double);
extern double logb(double);
extern double nextafter(double, double);
extern double pow(double, double);
extern double remainder(double, double);
extern double rint(double);
extern double scalb(double, double);
extern double sin(double);
extern double sinh(double);
extern double sqrt(double);
extern double tan(double);
extern double tanh(double);
extern double y0(double);
extern double y1(double);
extern double yn(int, double);
extern double drem(double, double);
extern float copysignf(float, float);
extern long double copysignl(long double, long double);
extern int finitef(float);
extern int finitel(long double);
extern float frexpf(float, int *);
extern long double ldexpf(float, int);
extern int finitef(float, int *);
extern float ldexpf(float, int);
extern float scalbf(float, float *);
extern float scalblf(float, long int);
extern float asinhf(float);
extern float atanhf(float);
extern float cbrtf(float);
extern float ceilf(float);
extern float cosf(float);
extern float coshf(float);

```c
extern double exp(double);
extern double expm1(double);
extern double fabs(double);
extern double floor(double);
extern double fmod(double, double);
extern double gamma(double);
extern double hypot(double, double);
extern int ilogb(double);
extern double j0(double);
extern double j1(double);
extern double jn(int, double);
extern double lgamma(double);
extern double log(double);
extern double log10(double);
extern double log1p(double);
extern double logb(double);
extern double nextafter(double, double);
extern double pow(double, double);
extern double remainder(double, double);
extern double rint(double);
extern double scalb(double, double);
extern double sin(double);
extern double sinh(double);
extern double sqrt(double);
extern double tan(double);
extern double tanh(double);
extern double y0(double);
extern double y1(double);
extern double yn(int, double);
extern double drem(double, double);
extern float copysignf(float, float);
extern long double copysignl(long double, long double);
extern int finitef(float);
extern int finitel(long double);
extern float frexpf(float, int *);
extern long double ldexpf(float, int);
extern int finitef(float, int *);
extern float ldexpf(float, int);
extern float scalbf(float, float *);
extern float scalblf(float, long int);
extern float asinhf(float);
extern float atanhf(float);
extern float cbrtf(float);
extern float ceilf(float);
extern float cosf(float);
extern float coshf(float);
```
extern long double coshl(long double);
extern long double cosl(long double);
extern float dremf(float, float);
extern long double dreml(long double, long double);
extern float erfcf(float);
extern long double erfcl(long double);
extern float erff(float);
extern long double erfl(long double);
extern double exp10(double);
extern float exp10f(float);
extern long double exp10l(long double);
extern double exp2(double);
extern float exp2f(float);
extern float expf(float);
extern long double expl(long double);
extern float expm1f(float);
extern long double expm1l(long double);
extern float fabsf(float);
extern long double fabsl(long double);
extern double fdim(double, double);
extern float fdimf(float, float);
extern long double fdiml(long double, long double);
extern float floorf(float);
extern long double floorl(long double);
extern double fma(double, double, double);
extern float fmaf(float, float, float);
extern long double fmal(long double, long double, long double);
extern double fmax(double, double);
extern float fmaxf(float, float);
extern long double fmaxl(long double, long double);
extern double fmin(double, double);
extern float fminf(float, float);
extern long double fminl(long double, long double);
extern float fmodf(float, float);
extern long double fmodl(long double, long double);
extern float gammaf(float);
extern long double gammal(long double);
extern float hypotf(float, float);
extern long double hypotl(long double, long double);
extern int ilogbf(float);
extern int ilogbl(long double);
extern float j0f(float);
extern long double j0l(long double);
extern float j1f(float);
extern long double j1l(long double);
extern float jnf(int, float);
extern long double jnl(int, long double);
extern double lgamma_r(double, int *);
extern float lgammaf(float);
extern float lgamma_r(float, int *);
extern double lgammal(long double);
extern long double lgammal_r(long double, int *);
extern long int l1rint(double);
extern long int l1rintf(float);
extern long int l1rintl(long double);
extern long int l1round(double);
extern long int l1roundf(float);
extern long int l1roundl(long double);
extern float log10f(float);
extern long double log10l(long double);
extern float log1pf(float);
extern long double log1pl(long double);
extern double log2(double);
extern float log2f(float);
extern long double log2l(long double);
extern float logbf(float);
extern long double logbl(long double);
extern float logf(float);
extern long double logl(long double);
extern long int lrint(double);
extern long int lrintf(float);
extern long int lrintl(long double);
extern long int lround(double);
extern long int lroundf(float);
extern long int lroundl(long double);
extern double nan(const char *);
extern float nanf(const char *);
extern long double nanl(const char *);
extern double nearbyint(double);
extern float nearbyintf(float);
extern long double nearbyintl(long double);
extern float nextafterf(float, float);
extern long double nextafterl(long double, long double);
extern float nexttoward(float, long double);
extern long double nexttowardl(long double, long double);
extern double pow10(double);
extern float pow10f(float);
extern long double pow10l(long double);
extern float powf(float, float);
extern long double powl(long double, long double);
extern float remainderf(float, float);
extern long double remainderl(long double, long double);
extern float remquo(double, double, int *);
extern float remquof(float, float, int *);
extern long double remquol(long double, long double, int *);
extern float rintf(float);
extern long double rintl(long double);
extern double round(double);
extern float roundf(float);
extern long double roundl(long double);
extern float scalbf(float, float);
extern long double scalbl(long double, long double);
extern float significandf(float);
extern long double significandl(long double);
extern void sincosf(float, float *, float *);
extern void sincosl(long double, long double *, long double *);
extern float sinf(float);
extern float sinhf(float);
extern float sinhl(long double);
extern float sqrtf(float);
extern float sqrtl(long double);
extern float tanf(float);
extern float tanhf(float);
extern float tanhl(long double);
extern double tgamma(double);
extern float tgammaf(float);
extern long double tgammal(long double);
extern float truncf(float);
extern long double truncl(long double);
extern float y0f(float);
extern long double y0l(long double);
extern float ylf(float);
extern long double y1l(long double);
extern float ynf(int, float);
extern long double ynl(int, long double);
13.8 Interface Definitions for libm

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

Other interfaces listed in Section 13.6 shall behave as described in the referenced base document.

__finite

Name
__finite — test for infinity

Synopsis
#include <math.h>
int __finite(double arg);

Description
__finite() has the same specification as isfinite() in ISO POSIX (2003), except that the argument type for __finite() is known to be double. __finite() is not in the source standard; it is only in the binary standard.

__finitef

Name
__finitef — test for infinity

Synopsis
#include <math.h>
int __finitef(float arg);

Description
__finitef() has the same specification as isfinite() in ISO POSIX (2003) except that the argument type for __finitef() is known to be float. __finitef() is not in the source standard; it is only in the binary standard.

__finitel

Name
__finitel — test for infinity

Synopsis
#include <math.h>
int __finitel(long double arg);

Description
__finitel() has the same specification as isfinite() in the ISO POSIX (2003), except that the argument type for __finitel() is known to be long double. __finitel() is not in the source standard; it is only in the binary standard.
__fpclassify

Name
__fpclassify — Classify real floating type

Synopsis

int __fpclassify(double arg);

Description
__fpclassify() has the same specification as fpclassify() in ISO POSIX (2003), except that the argument type for __fpclassify() is known to be double.
__fpclassify() is not in the source standard; it is only in the binary standard.

__fpclassifyf

Name
__fpclassifyf — Classify real floating type

Synopsis

int __fpclassifyf(float arg);

Description
__fpclassifyf() has the same specification as fpclassify() in ISO POSIX (2003), except that the argument type for __fpclassifyf() is known to be float.
__fpclassifyf() is not in the source standard; it is only in the binary standard.

__signbit

Name
__signbit — test sign of floating point value

Synopsis

#include <math.h>
int __signbit(double arg);

Description
__signbit() has the same specification as signbit() in ISO POSIX (2003), except that the argument type for __signbit() is known to be double.
__signbit() is not in the source standard; it is only in the binary standard.
__signbitf

Name
__signbitf — test sign of floating point value

Synopsis
#include <math.h>
int __signbitf(float arg);

Description
__signbitf() has the same specification as signbit() in ISO POSIX (2003), except that the argument type for __signbitf() is known to be float.
__signbitf() is not in the source standard; it is only in the binary standard.

clog10

Name
clog10 — Logarithm of a Complex Number

Synopsis
#include <complex.h>
double complex clog10(double complex z);

Description
The clog10() function shall compute the base 10 logarithm of the complex number z.

Return Value
The clog10() function shall return the base 10 logarithm.

clog10f

Name
clog10f — Logarithm of a Complex Number

Synopsis
#include <complex.h>
float complex clog10f(float complex z);

Description
The clog10f() function shall compute the base 10 logarithm of the complex number z.

Return Value
The clog10f() function shall return the base 10 logarithm.
clog10l

**Name**
clog10l — Logarithm of a Complex Number

**Synopsis**
```c
#include <complex.h>
long double complex clog10l(long double complex z);
```

**Description**
The `clog10l()` function shall compute the base 10 logarithm of the complex number `z`.

**Return Value**
The `clog10l()` function shall return the base 10 logarithm.

drem

**Name**
drem — Floating Point Remainder (DEPRECATED)

**Synopsis**
```c
#include <math.h>
double drem(double x, double y);
```

**Description**
The `drem()` function shall return the floating point remainder, `x` REM `y` as required by IEC 60559/IEEE 754 Floating Point in the same way as `remainder()`.

**Note**: This function is included only for backwards compatibility; applications should use `remainder()` instead.

**Returns**
See `remainder()`.

**See Also**
`remainder()`, `dremf()`, `dreml()`
dremf

Name

dremf — Floating Point Remainder (DEPRECATED)

Synopsis

#include <math.h>
double dremf(double x, double y);

Description

The dremf() function shall return the floating point remainder, \( x \text{ REM } y \) as required by IEC 60559/IEEE 754 Floating Point in the same way as remainderf().

Note: This function is included only for backwards compatibility; applications should use remainderf() instead.

Returns

See remainderf().

See Also

remainderf(), drem(), dreml()

dreml

Name

dreml — Floating Point Remainder (DEPRECATED)

Synopsis

#include <math.h>
double dreml(double x, double y);

Description

The dreml() function shall return the floating point remainder, \( x \text{ REM } y \) as required by IEC 60559/IEEE 754 Floating Point in the same way as remainderl().

Note: This function is included only for backwards compatibility; applications should use remainderl() instead.

Returns

See remainderl().

See Also

remainderl(), drem(), dremf()
exp10

Name
exp10 — Base-10 power function

Synopsis
#include <math.h>
double exp10(double x);

Description
The exp10() function shall return 10^x.

Note: This function is identical to pow10().

Returns
Upon successful completion, exp10() shall return 10 rised to the power of x.
If the correct value would cause overflow, a range error shall occur and
exp10() shall return ±HUGE_VAL, with the same sign as the correct value of
the function.

See Also
pow10(), exp10f(), exp10l()

exp10f

Name
exp10f — Base-10 power function

Synopsis
#include <math.h>
float exp10f(float x);

Description
The exp10f() function shall return 10^x.

Note: This function is identical to pow10f().

Returns
Upon successful completion, exp10f() shall return 10 rised to the power of x.
If the correct value would cause overflow, a range error shall occur and
exp10f() shall return ±HUGE_VALF, with the same sign as the correct value of
the function.

See Also
pow10f(), exp10(), exp10l()
exp10l

Name
exp10l — Base-10 power function

Synopsis
#include <math.h>
long double exp10l(long double x);

Description
The exp10l() function shall return 10^x.

Note: This function is identical to pow10l().

Returns
Upon successful completion, exp10l() shall return 10 rised to the power of x.
If the correct value would cause overflow, a range error shall occur and exp10l() shall return ±HUGE_VALL, with the same sign as the correct value of the function.

See Also
pow10l(), exp10(), exp10f()

fedisableexcept

Name
fedisableexcept — disable floating point exceptions

Synopsis
#include <fenv.h>
int fedisableexcept(int excepts);

Description
The fedisableexcept() function disables traps for each of the exceptions represented by the mask excepts.

Return Value
The fedisableexcept() function returns the previous set of enabled exceptions on success. On error, -1 is returned.

Errors
No errors are defined, but the function will fail if not supported on the architecture.
feenableexcept

Name
feenableexcept — enable floating point exceptions

Synopsis
#include <fenv.h>
int feenableexcept(int excepts);

Description
The feenableexcept() function enables traps for each of the exceptions represented by the mask excepts.

Return Value
The feenableexcept() function returns the previous set of enabled exceptions on success. On error, -1 is returned.

Errors
No errors are defined, but the function will fail if not supported on the architecture.

fegetexcept

Name
fegetexcept — query floating point exception handling state

Synopsis
#include <fenv.h>
int fegetexcept

Description
The fegetexcept() function returns the set of all currently enabled exceptions.

Return Value
The mremap() function returns the set of all currently enabled exceptions on success. On error, -1 is returned.

Errors
No errors are defined, but the function will fail if not supported on the architecture.

finite

Name
finite — test for infinity (DEPRECATED)

Synopsis
#define _SVID_SOURCE


#include <math.h>
int finite(double arg);

Description
The `finite()` function shall test whether its argument is neither INFINITY nor not a number (NaN).

Returns
On success, `finite()` shall return 1. Otherwise the function shall return 0.

Note: The ISO C (1999) standard defines the function `isfinite()`, which is more general purpose. The `finite()` function is deprecated, and applications should use `isfinite()` instead. A future revision of this standard may remove this function.

See Also
`isfinite()`, `finitef()`, `finitel()`

finitef

Name
finitef — test for infinity (DEPRECATED)

Synopsis
#define _SVID_SOURCE
#include <math.h>
int finitef(float arg);

Description
The `finitef()` function shall test whether its argument is neither INFINITY nor not a number (NaN).

Returns
On success, `finitef()` shall return 1. Otherwise the function shall return 0.

Note: The ISO C (1999) standard defines the function `isfinite()`, which is more general purpose. The `finitef()` function is deprecated, and applications should use `isfinite()` instead. A future revision of this standard may remove this function.

See Also
`isfinite()`, `finite()`, `finitel()`

finitel

Name
finitel — test for infinity (DEPRECATED)

Synopsis
#define _SVID_SOURCE
#include <math.h>
int finitel(long double arg);

**Description**
The `finitel()` function shall test whether its argument is neither INFINITY nor not a number (NaN).

**Returns**
On success, `finitel()` shall return 1. Otherwise the function shall return 0.

*Note:* The ISO C (1999) standard defines the function `isfinite()`, which is more general purpose. The `finitel()` function is deprecated, and applications should use `isfinite()` instead. A future revision of this standard may remove this function.

**See Also**
`isfinite()`, `finite()`, `finitef()`

**gamma**

**Name**
gamma — log gamma function (DEPRECATED)

**Synopsis**

```
#include <math.h>
double gammaf(double x);
```

**Description**
The `gamma()` function is identical to `lgamma()` in ISO POSIX (2003).

*Note:* The name `gamma()` for this function is deprecated and should not be used.

**Returns**
See `lgamma()`.

**See Also**
`lgamma()`, `lgammaf()`, `lgammal()`, `gammaf()`, `gammal()`
**gammaf**

**Name**

gammaf — log gamma function (DEPRECATED)

**Synopsis**

```c
#include <math.h>
float gammaf(float x);
```

**Description**

The `gammaf()` function is identical to `lgammaf()` in ISO POSIX (2003).

**Note:** The name `gammaf()` for this function is deprecated and should not be used.

**Returns**

See `lgammaf()`.

**See Also**

`lgamma()`, `lgammaf()`, `lgammal()`, `gamma()`, `gammal()`

**gammal**

**Name**

`gammal` — log gamma function (DEPRECATED)

**Synopsis**

```c
#include <math.h>
long double gammal(long double x);
```

**Description**

The `gammal()` function is identical to `lgammal()` in ISO POSIX (2003).

**Note:** The name `gammal()` for this function is deprecated and should not be used.

**Returns**

See `lgammal()`.

**See Also**

`lgamma()`, `lgammaf()`, `lgammal()`, `gamma()`, `gammal()`
\textbf{\texttt{j0f}}

\textbf{Name}
\texttt{j0f} — Bessel functions

\textbf{Synopsis}
\begin{verbatim}
#include <math.h>
float j0f(float x);
\end{verbatim}

\textbf{Description}
The \texttt{j0f()} function is identical to \texttt{j0()}, except that the argument $x$ and the return value is a float.

\textbf{Returns}
See \texttt{j0()}. 

\textbf{See Also}
\texttt{j0()}, \texttt{j0l()}, \texttt{j1()}, \texttt{j1f()}, \texttt{j1l()}, \texttt{jn()}, \texttt{jnf()}, \texttt{jnl()}, \texttt{y0()}, \texttt{y0f()}, \texttt{y0l()}, \texttt{yl()}, \texttt{y1f()}, \texttt{y1l()}, \texttt{yn()}, \texttt{ynf()}, \texttt{ynl()}

\textbf{\texttt{j0l}}

\textbf{Name}
\texttt{j0l} — Bessel functions

\textbf{Synopsis}
\begin{verbatim}
#include <math.h>
long double j0l(long double x);
\end{verbatim}

\textbf{Description}
The \texttt{j0l()} function is identical to \texttt{j0()}, except that the argument $x$ and the return value is a long double.

\textbf{Returns}
See \texttt{j0()}. 

\textbf{See Also}
\texttt{j0()}, \texttt{j0f()}, \texttt{j1()}, \texttt{j1f()}, \texttt{j1l()}, \texttt{jn()}, \texttt{jnf()}, \texttt{jnl()}, \texttt{y0()}, \texttt{y0f()}, \texttt{y0l()}, \texttt{yl()}, \texttt{y1f()}, \texttt{y1l()}, \texttt{yn()}, \texttt{ynf()}, \texttt{ynl()}

© 2007 Linux Foundation
j1f

Name
j1f — Bessel functions

Synopsis
#include <math.h>
float j1f(float x);

Description
The j1f() function is identical to j1(), except that the argument x and the return value is a float.

Returns
See j1().

See Also
j0(), j0f(), j0l(), j1(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()

j1l

Name
j1l — Bessel functions

Synopsis
#include <math.h>
long double j1l(long double x);

Description
The j1l() function is identical to j1(), except that the argument x and the return value is a long double.

Returns
See j0().

See Also
j0(), j0f(), j0l(), j1(), j1f(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()
jnf

Name
jnf — Bessel functions

Synopsis
#include <math.h>
float jnf(float x);

Description
The jnf() function is identical to jn(), except that the argument x and the return value is a float.

Returns
See jn().

See Also
j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()

jnl

Name
jnl — Bessel functions

Synopsis
#include <math.h>
long double jnl(long double x);

Description
The jnl() function is identical to jn(), except that the argument x and the return value is a long double.

Returns
See jn().

See Also
j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()
lgamma_r

Name

lgamma_r — log gamma functions

Synopsis

#include <math.h>
double lgamma_r(double x, int * signp);

Description

The lgamma_r() function shall compute the natural logarithm of the absolute value of the Gamma function, as lgamma(). However, instead of setting the external integer signgam to the sign of the Gamma function, lgamma_r() shall set the integer referenced by signp to the sign.

Returns

See lgamma() and signgam.

See Also

lgamma(), lgammaf_r(), lgammal_r(), signgam

lgammaf_r

Name

lgammaf_r — log gamma functions

Synopsis

#include <math.h>
float lgammaf_r(float x, int * signp);

Description

The lgammaf_r() function shall compute the natural logarithm of the absolute value of the Gamma function, as lgammaf(). However, instead of setting the external integer signgam to the sign of the Gamma function, lgammaf_r() shall set the integer referenced by signp to the sign.

Returns

See lgammaf() and signgam.

See Also

lgamma(), lgamma_r(), lgammal_r(), signgam
lgammal_r

**Name**

lgammal_r — log gamma functions

**Synopsis**

```c
#include <math.h>
double lgammal_r(double x, int * signp);
```

**Description**

The lgammal_r() function shall compute the natural logarithm of the absolute value of the Gamma function, as lgammal(). However, instead of setting the external integer signgam to the sign of the Gamma function, lgammal_r() shall set the integer referenced by signp to the sign.

**Returns**

See lgammal() and signgam.

**See Also**

lgamma(), lgamma_r(), lgammaf_r(), signgam

pow10

**Name**

pow10 — Base-10 power function

**Synopsis**

```c
#include <math.h>
double pow10(double x);
```

**Description**

The pow10() function shall return $10^x$.

**Note:** This function is identical to exp10().

**Returns**

Upon successful completion, pow10() shall return 10 raised to the power of $x$.

If the correct value would cause overflow, a range error shall occur and pow10() shall return ±HUGE_VAL, with the same sign as the correct value of the function.

**See Also**

exp10(), pow10f(), pow10l()
pow10f

Name
pow10f — Base-10 power function

Synopsis
#include <math.h>
float pow10f(float x);

Description
The pow10f() function shall return 10^x.

Note: This function is identical to exp10f().

Returns
Upon successful completion, pow10f() shall return 10 rised to the power of x.
If the correct value would cause overflow, a range error shall occur and pow10f() shall return ±HUGE_VALF, with the same sign as the correct value of the function.

See Also
exp10f(), pow10(), pow10l()

pow10l

Name
pow10l — Base-10 power function

Synopsis
#include <math.h>
long double pow10l(long double x);

Description
The pow10l() function shall return 10^x.

Note: This function is identical to exp10l().

Returns
Upon successful completion, pow10l() shall return 10 rised to the power of x.
If the correct value would cause overflow, a range error shall occur and pow10l() shall return ±HUGE_VALL, with the same sign as the correct value of the function.

See Also
exp10l(), pow10(), pow10f()
significand

Name
significand — floating point mantissa

Synopsis

#include <math.h>
double significand(double x);

Description
The significand() function shall return the mantissa of \( x \), \( \text{sig} \) such that \( x = \text{sig} \times 2^n \) scaled such that \( 1 \leq \text{sig} < 2 \).

Note: This function is intended for testing conformance to IEC 60559/IEEE 754 Floating Point, and its use is not otherwise recommended.

Returns
Upon successful completion, significand() shall return the mantissa of \( x \) in the range \( 1 \leq \text{sig} < 2 \).
If \( x \) is 0, ±HUGE_VAL, or NaN, the result is undefined.

See Also
significandf(), significandl()

significandf

Name
significandf — floating point mantissa

Synopsis

#include <math.h>
float significandf(float x);

Description
The significandf() function shall return the mantissa of \( x \), \( \text{sig} \) such that \( x = \text{sig} \times 2^n \) scaled such that \( 1 \leq \text{sig} < 2 \).

Note: This function is intended for testing conformance to IEC 60559/IEEE 754 Floating Point, and its use is not otherwise recommended.

Returns
Upon successful completion, significandf() shall return the mantissa of \( x \) in the range \( 1 \leq \text{sig} < 2 \).
If \( x \) is 0, ±HUGE_VALF, or NaN, the result is undefined.

See Also
significand(), significandl()
**significandl**

**Name**

significandl — floating point mantissa

**Synopsis**

```c
#include <math.h>
long double significandl(long double x);
```

**Description**

The `significandl()` function shall return the mantissa of `x`, `sig` such that `x = sig * 2^n` scaled such that `1 \leq sig < 2`.

**Note:** This function is intended for testing conformance to IEC 60559/IEEE 754 Floating Point, and its use is not otherwise recommended.

This function is equivalent to `scalb(x, (double)-ilogb(x))`.

**Returns**

Upon successful completion, `significandl()` shall return the mantissa of `x` in the range `1 \leq sig < 2`.

If `x` is 0, ±HUGE_VALL, or NaN, the result is undefined.

**See Also**

`significand()`, `significandf()`

**sincos**

**Name**

sincos — trigonometric functions

**Synopsis**

```c
#define __GNU_SOURCE
#include <math.h>
void sincos(double x, double *sin, double *cos);
```

**Description**

The `sincos()` function shall calculate both the sine and cosine of `x`. The sine shall be stored in the location referenced by `sin`, and the cosine in the location referenced by `cosine`.

**Returns**

None. See `sin()` and `cos()` for possible error conditions.

**See Also**

`cos()`, `sin()`, `sincosf()`, `sincosl()`
sincosf

Name

sincosf — trigonometric functions

Synopsis

#define _GNU_SOURCE
#include <math.h>
void sincosf(float x, float * sin, float * cos);

Description

The sincosf() function shall calculate both the sine and cosine of x. The sine
shall be stored in the location referenced by sin, and the cosine in the location
referenced by cosine.

Returns

None. See sin() and cos() for possible error conditions.

See Also

cos(), sin(), sincos(), sincosl()

sincosl

Name

sincosl — trigonometric functions

Synopsis

#define _GNU_SOURCE
#include <math.h>
void sincosl(long double x, long double * sin, long double * cos);

Description

The sincosl() function shall calculate both the sine and cosine of x. The sine
shall be stored in the location referenced by sin, and the cosine in the location
referenced by cosine.

Returns

None. See sin() and cos() for possible error conditions.

See Also

cos(), sin(), sincos(), sincosl()
y0f

Name
y0f — Bessel functions

Synopsis
#include <math.h>
float y0f(float x);

Description
The y0f() function is identical to y0(), except that the argument x and the return value is a float.

Returns
See y0().

See Also
j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()

y0l

Name
y0l — Bessel functions

Synopsis
#include <math.h>
long double y0l(long double x);

Description
The y0l() function is identical to y0(), except that the argument x and the return value is a long double.

Returns
See y0().

See Also
j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()
y1f

Name
y1f — Bessel functions

Synopsis
#include <math.h>
float y1f(float x);

Description
The y1f() function is identical to y1(), except that the argument x and the return value is a float.

Returns
See y1().

See Also
j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1l(), yn(), ynf(), ynl()

y1l

Name
y1l — Bessel functions

Synopsis
#include <math.h>
long double y1l(long double x);

Description
The y1l() function is identical to y1(), except that the argument x and the return value is a long double.

Returns
See j0().

See Also
j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), yn(), ynf(), ynl()
ynf

Name

ynf — Bessel functions

Synopsis

#include <math.h>
float ynf(float x);

Description

The ynf() function is identical to yn(), except that the argument x and the return value is a float.

Returns

See yn().

See Also

j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), yl(), ylf(), yll(), yn(), ynl()

ynl

Name

ynl — Bessel functions

Synopsis

#include <math.h>
long double ynl(long double x);

Description

The ynl() function is identical to yn(), except that the argument x and the return value is a long double.

Returns

See yn().

See Also

j0(), j0f(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), yl(), ylf(), yll(), yn(), ynl()

13.9 Interfaces for libpthread

Table 13-37 defines the library name and shared object name for the libpthread library

Table 13-37 libpthread Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libpthread</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libpthread.so.0</td>
</tr>
</tbody>
</table>
The behavior of the interfaces in this library is specified by the following specifications:

[LFS] Large File Support
[LSB] This Specification

13.9.1 Realtime Threads

13.9.1.1 Interfaces for Realtime Threads
An LSB conforming implementation shall provide the generic functions for Realtime Threads specified in Table 13-38, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-38 libpthread - Realtime Threads Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getinheritsched</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getschedpolicy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getscope</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getinheritsched</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setschedpolicy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setscope</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_setschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_setschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getcpulockid</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

13.9.2 Advanced Realtime Threads

13.9.2.1 Interfaces for Advanced Realtime Threads
An LSB conforming implementation shall provide the generic functions for Advanced Realtime Threads specified in Table 13-39, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-39 libpthread - Advanced Realtime Threads Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_barrier_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_barrier_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_barrier_wait</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_barrierattrttr_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_barriertaattrttr_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_barriertaattrttr_setpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getcpuclockid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_spin_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_spin_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_spin_locallock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_spin_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_spin_unlock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_spin_unlock</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>
13.9.3 Posix Threads

13.9.3.1 Interfaces for Posix Threads

An LSB conforming implementation shall provide the generic functions for Posix Threads specified in Table 13-40, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-40 libpthread - Posix Threads Function Interfaces

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Implementation</th>
<th>Function Name</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_get guardsize [SUSv3]</td>
<td></td>
<td>pthread_attr_get schedparam [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_get stacksize [SUSv3]</td>
<td></td>
<td>pthread_attr_set detachstate [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_sets schedparam [SUSv3]</td>
<td></td>
<td>pthread_attr_sets stack [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_sets stackaddr [SUSv3]</td>
<td></td>
<td>pthread_attr_sets stacksize [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_cancel [SUSv3]</td>
<td></td>
<td>pthread_cond_broadcast [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_cond_signal [SUSv3]</td>
<td></td>
<td>pthread_cond_destroy [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_cond_timedwait [SUSv3]</td>
<td></td>
<td>pthread_cond_wait [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_condattr_getpshared [SUSv3]</td>
<td></td>
<td>pthread_condattr_init [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_detach [SUSv3]</td>
<td></td>
<td>pthread_equal [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getspecific [SUSv3]</td>
<td></td>
<td>pthread_exit [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getconcurrency [SUSv3]</td>
<td></td>
<td>pthread_getconcurrency [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_join [SUSv3]</td>
<td></td>
<td>pthread_key_create [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_key_delete [SUSv3]</td>
<td></td>
<td>pthread_mutex_init [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_timedlock [SUSv3]</td>
<td></td>
<td>pthread_mutex_unlock [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_getpshared [SUSv3]</td>
<td></td>
<td>pthread_mutexattr_init [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_settype [SUSv3]</td>
<td></td>
<td>pthread_mutexattr_setpshared [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_setgetpshared</td>
<td></td>
<td>pthread_mutexattr_getpshared [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_settype [SUSv3]</td>
<td></td>
<td>pthread_mutexattr_settype [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_gettype [SUSv3]</td>
<td></td>
<td>pthread_mutexattr_settype [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_destroy [SUSv3]</td>
<td></td>
<td>pthread_rwlock_init [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_rdlock [SUSv3]</td>
<td></td>
<td>pthread_rwlock_timedlock [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_timedlock [SUSv3]</td>
<td></td>
<td>pthread_rwlock_trylock [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_rdlock [SUSv3]</td>
<td></td>
<td>pthread_rwlock_timedlock [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_timedlock [SUSv3]</td>
<td></td>
<td>pthread_rwlock_trylock [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_getshared [SUSv3]</td>
<td></td>
<td>pthread_rwlock_gettype [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_settype [SUSv3]</td>
<td></td>
<td>pthread_rwlock_init [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_settype [SUSv3]</td>
<td></td>
<td>pthread_rwlock_destroy [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_getshared [SUSv3]</td>
<td></td>
<td>pthread_rwlock_settype [SUSv3]</td>
<td></td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for Posix Threads specified in Table 13-41, with the full mandatory functionality as described in the referenced underlying specification.

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

Table 13-41 libpthread - Posix Threads Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getstackaddr [SUSv3]</td>
<td>pthread_attr_setsstackaddr [SUSv3]</td>
<td></td>
</tr>
</tbody>
</table>

13.9.4 Thread aware versions of libc interfaces

13.9.4.1 Interfaces for Thread aware versions of libc interfaces

An LSB conforming implementation shall provide the generic functions for Thread aware versions of libc interfaces specified in Table 13-42, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-42 libpthread - Thread aware versions of libc interfaces Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>lseek64 [LFS]</td>
<td>open64 [LFS]</td>
<td>pread [SUSv3]</td>
<td>pread64 [LFS]</td>
</tr>
<tr>
<td>pwrite [SUSv3]</td>
<td>pwrite64 [LFS]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.10 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and
application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

13.10.1 pthread.h

```c
#define PTHREAD_MUTEX_DEFAULT   0
#define PTHREAD_MUTEX_NORMAL    0
#define PTHREAD_SCOPE_SYSTEM    0
#define PTHREAD_MUTEX_RECURSIVE 1
#define PTHREAD_SCOPE_PROCESS   1
#define PTHREAD_MUTEX_ERRORCHECK 2
#define PTHREAD_RWLOCK_DEFAULT_NP 2
#define __SIZEOF_PTHREAD_BARRIERATTR_T 4
#define PTHREAD_MUTEX_INITIALIZER \
{0,0,PTHREAD_MUTEX_NORMAL,0,0,PTHREAD_MUTEX_NORMAL,0,NP,0,0,\} \ \{0,0,PTHREAD_MUTEX_NORMAL,0,0,PTTHREAD_MUTEX_NORMAL,0,\} \
PTTHREAD_PROCESS_PRIVATE ) 
#define pthread_cleanup_push(routine,arg) \ 
{struct _pthread_cleanup_buffer _buffer;\_pthread_cleanup_push(&_buffer,(routine),(arg));\} 
#define pthread_cleanup_pop(execute) \ 
{struct _pthread_cleanup_buffer _buffer;\_pthread_cleanup_pop(&_buffer,(execute));}\} 
#define __LOCK_INITIALIZER      { 0, 0 } 
#define PTHREAD_COND_INITIALIZER        {__LOCK_INITIALIZER,0} 
struct _pthread_cleanup_buffer { 
  void (*__routine) (void *); 
  void *__arg; 
  int __canceltype; 
  struct _pthread_cleanup_buffer *__prev; 
}; 
typedef unsigned int pthread_key_t; 
typedef int pthread_once_t; 
typedef long long int __pthread_cond_align_t; 
typedef volatile int pthread_spinlock_t; 
typedef union { 
char __size[__SIZEOF_PTHREAD_BARRIERATTR_T]; 
int __align; 
} pthread_barrierattr_t; 
typedef unsigned long int pthread_t; 
struct _pthread_fastlock { 
  long int __status; 
  int __spinlock; 
}; 
typedef struct _pthread_descr_struct *pthread_descr; 
typedef struct { 
  int __m_reserved; 
  int __m_count; 
  _pthread_descr __m_owner; 
  int __m_kind; 
  struct _pthread_fastlock __m_lock; 
} pthread_mutex_t; 
typedef struct { 
```
int __mutexkind;
) pthread_mutexattr_t;

typedef struct {
  int __detachstate;
  int __schedpolicy;
  struct sched_param __schedparam;
  int __inheritsched;
  int __scope;
  size_t __guardsize;
  int __stackaddr_set;
  void *__stackaddr;
  unsigned long int __stacksize;
) 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 __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

extern int pthread_barrier_destroy(pthread_barrier_t *);
extern int pthread_barrier_init(pthread_barrier_t *,
                              const pthread_barrierattr_t *,
                              unsigned int);
extern int pthread_barrier_wait(pthread_barrier_t *);
extern int pthread_barrierattr_destroy(pthread_barrierattr_t *);
extern int pthread_barrierattr_init(pthread_barrierattr_t *);
extern int pthread_barrierattr_setpshared(pthread_barrierattr_t *,
                            int);
extern int pthread_getcpuclockid(pthread_t, clockid_t *);
extern int pthread_spin_destroy(pthread_spinlock_t *);
extern int pthread_spin_init(pthread_spinlock_t *, int);
extern int pthread_spin_lock(pthread_spinlock_t *);
extern int pthread_spin_unlock(pthread_spinlock_t *);
extern int pthread_mutex_timedlock(pthread_mutex_t *,
        const struct timespec *);

extern int pthread_barrierattr_getpshared(const pthread_barrierattr_t *,
        int *);
extern void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *,
        int);
extern void _pthread_cleanup_push(struct _pthread_cleanup_buffer *,
        void (*__pshared) (void *)
        , void *);

extern int pthread_attr_destroy(pthread_attr_t *);
extern int pthread_attr_getdetachstate(const pthread_attr_t *,
        int *);
extern int pthread_attr_getinheritsched(const pthread_attr_t *,
        int *);
extern int pthread_attr_getschedparam(const pthread_attr_t *,
        struct sched_param *);

extern int pthread_attr_getschedpolicy(const pthread_attr_t *,
        int *);
extern int pthread_attr_getscope(const pthread_attr_t *,
        int *);
extern int pthread_attr_init(pthread_attr_t *);
extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
extern int pthread_attr_setschedparam(pthread_attr_t *,
        int *, const struct sched_param *);

extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
extern int pthread_attr_setscope(pthread_attr_t *, int);
extern int pthread_cancel(pthread_t);
extern int pthread_cond_broadcast(pthread_cond_t *);
extern int pthread_cond_destroy(pthread_cond_t *);
extern int pthread_cond_init(pthread_cond_t *,
        const pthread_condattr_t *);
extern int pthread_cond_signal(pthread_cond_t *);
extern int pthread_cond_timedwait(pthread_cond_t *,
        pthread_mutex_t *,
        const struct timespec *);

extern int pthread_cond_wait(pthread_cond_t *,
        pthread_mutex_t *);
extern int pthread_condattr_destroy(pthread_condattr_t *);
extern int pthread_condattr_init(pthread_condattr_t *);

extern int pthread_create(pthread_t *, const pthread_attr_t *,
        void *(*__pshared) (void *)
        , void *);
extern int pthread_detach(pthread_t);
extern int pthread_equal(pthread_t, pthread_t);
extern void pthread_exit(void *);

extern int pthread_getschedparam(pthread_t, int *, struct sched_param *);
extern void *pthread_getspecific(pthread_key_t);
extern int pthread_join(pthread_t, void **);
extern int pthread_key_create(pthread_key_t *, void (*__pshared) (void *)
        , void *);

extern int pthread_key_delete(pthread_key_t);
extern int pthread_mutex_destroy(pthread_mutex_t *);
extern int pthread_mutex_init(pthread_mutex_t *,
        const pthread_mutexattr_t *);

extern int pthread_mutex_lock(pthread_mutex_t *);
extern int pthread_mutex_trylock(pthread_mutex_t *);
extern int pthread_mutex_unlock(pthread_mutex_t *);

© 2007 Linux Foundation
extern int pthread_mutexattr_destroy(pthread_mutexattr_t *);
extern int pthread_mutexattr_init(pthread_mutexattr_t *);
extern int pthread_once(pthread_once_t *, void (*__pshared)(void));
extern int pthread_rwlock_destroy(pthread_rwlock_t *);
extern int pthread_rwlock_init(pthread_rwlock_t *,
        const pthread_rwlockattr_t *);
extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *);
extern int pthread_rwlock_trywrlock(pthread_rwlock_t *);
extern int pthread_rwlock_unlock(pthread_rwlock_t *);
extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
extern int pthread_rwlockattr_getpshared(const pthread_rwlockattr_t *,
        int *);
extern int pthread_rwlockattr_init(pthread_rwlockattr_t *, int);
extern pthread_t pthread_self(void);
extern int pthread_setcancelstate(int, int *);
extern int pthread_setcanceltype(int, int *);
extern int pthread_setschedparam(pthread_t, int, const struct sched_param *);
extern int pthread_setspecific(pthread_key_t, const void *);
extern int pthread_attr_getguardsize(const pthread_attr_t *, size_t *);
extern int pthread_attr_setguardsize(pthread_attr_t *, size_t);
extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
extern int pthread_attr_setstackaddr(const pthread_attr_t *, void *
        *
    , void *);
extern int pthread_attr_getstackaddr(const pthread_attr_t *, void **
        *
    , void *);
extern int pthread_mutexattr_gettype(const pthread_mutexattr_t *,
        int *);
extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
extern int pthread_attr_getstack(const pthread_attr_t *, void *);
extern int pthread_attr_setstack(pthread_attr_t *, void *
        *
    , size_t *
    , void *);
extern int pthread_attr_setstackaddr(pthread_attr_t *, void *
        *
    , size_t *
    , void *);
extern int pthread_mutexattr_setpshared(const pthread_mutexattr_t *,
        int *);
extern int pthread_mutexattr_getpshared(const pthread_mutexattr_t *,
        int *);
extern int pthread_attr_setpshared(pthread_attr_t *, int *);
extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *,
        const struct timespec *);
extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *,
        const struct timespec *);
extern int __register_atfork(void (*__pshared)(void),
        void (*__pshared)(void),
        void (*__pshared)(void)
        , void *);
### 13.10.2 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))

extern int sem_close(sem_t *);
extern int sem_destroy(sem_t *);
extern int sem_getvalue(sem_t *, int *);
extern int sem_init(sem_t *, int, unsigned int);
extern sem_t *sem_open(const char *, int, ...);
extern int sem_post(sem_t *);
extern int sem_trywait(sem_t *);
extern int sem_unlink(const char *);
extern int sem_wait(sem_t *);
extern int sem_timedwait(sem_t *, const struct timespec *);

### 13.11 Interface Definitions for libpthread

The interfaces defined on the following pages 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 in Section 13.9 shall behave as described in the referenced base document.

**_pthread_cleanup_pop**

**Name**

_pthread_cleanup_pop — establish cancellation handlers

**Synopsis**

```c
#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.

13.12 Interfaces for libgcc_s

Table 13-43 defines the library name and shared object name for the libgcc_s
library

Table 13-43 libgcc_s Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libgcc_s</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libgcc_s.so.1</td>
</tr>
</tbody>
</table>

13.12.1 Unwind Library

13.12.1.1 Interfaces for Unwind Library

No external functions are defined for libgcc_s - Unwind Library in this part of
the specification. See also the relevant architecture specific part of this
specification.

13.13 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.
Where an interface is defined as requiring a particular system header file all of
the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not
to repeat source interface definitions available elsewhere. System providers and
application developers should use this ABI to supplement - not to replace -
source interface definition specifications.

This specification uses the ISO C (1999) 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.
13.13.1 unwind.h

struct _Unwind_Context;

typedef void * _Unwind_Ptr;
typedef unsigned int _Unwind_Word;
typedef u_int64_t _Unwind_Exception_Class;

typedef enum {
    _URC_NO_REASON = 0,
    _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;

typedef void (*_Unwind_Exception_Cleanup_Fn) (enum,
                                             struct
                                             _Unwind_Exception *);

struct _Unwind_Exception {
    u_int64_t exception_class;
    _Unwind_Exception_Cleanup_Fn exception_cleanup;
    u_int64_t private_1;
    u_int64_t private_2;
};

#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

typedef int _Unwind_Action;

13.14 Interfaces for libdl

Table 13-44 defines the library name and shared object name for the libdl library

<table>
<thead>
<tr>
<th>Library:</th>
<th>libdl</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libdl.so.2</td>
</tr>
</tbody>
</table>

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

[LSB] This Specification

13.14.1 Dynamic Loader

13.14.1.1 Interfaces for Dynamic Loader

An LSB conforming implementation shall provide the generic functions for Dynamic Loader specified in Table 13-45, with the full mandatory functionality as described in the referenced underlying specification.
13 Base Libraries

ISO/IEC 23360 Part 1:2007(E)

Table 13-45 libdl - Dynamic Loader Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
</table>

13.15 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

13.15.1 dlfcn.h

```c
#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;

extern int dladdr(const void *, Dl_info *);
extern int dlclose(void *);
extern char *dlerror(void);
extern void *dlopen(const char *, int);
extern void *dlsym(void *, const char *);
```

13.16 Interface Definitions for libdl

The interfaces defined on the following pages 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 in Section 13.14 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;
}
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`. This address has type "pointer to type", where *type* is the type of the symbol `dli_sname`.

  **Example:** If the symbol in `dli_sname` is a function, then the type of `dli_saddr` is of type "pointer to function".

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 `dlerror()`.

Errors

See `dlerror()`.

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
The dlopen() function 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.

Note: 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).

13.17 Interfaces for librt

Table 13-46 defines the library name and shared object name for the librt library

Table 13-46 librt Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>librt</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>librt.so.1</td>
</tr>
</tbody>
</table>
The behavior of the interfaces in this library is specified by the following specifications:


13.17.1 Shared Memory Objects

13.17.1.1 Interfaces for Shared Memory Objects

An LSB conforming implementation shall provide the generic functions for Shared Memory Objects specified in Table 13-47, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-47 librt - Shared Memory Objects Function Interfaces

| shm_open [SUSv3] | shm_unlink [SUSv3] |

13.17.2 Clock

13.17.2.1 Interfaces for Clock

An LSB conforming implementation shall provide the generic functions for Clock specified in Table 13-48, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-48 librt - Clock Function Interfaces


13.17.3 Timers

13.17.3.1 Interfaces for Timers

An LSB conforming implementation shall provide the generic functions for Timers specified in Table 13-49, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-49 librt - Timers Function Interfaces


13.17.4 Message Queues

13.17.4.1 Interfaces for Message Queues

An LSB conforming implementation shall provide the generic functions for Message Queues specified in Table 13-50, with the full mandatory functionality as described in the referenced underlying specification.
Table 13-50 librt - Message Queues Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>GLIBC_2.3.4</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>mq_close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_getattr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_notify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_open</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_receive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_send</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_setattr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_timedreceive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq.unlink</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.18 Data Definitions for librt

This section defines global identifiers and their values that are associated with interfaces contained in librt. 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

13.18.1 mqueue.h

typedef int mqd_t;
struct mq_attr {
    long int mq_flags;
    long int mq_maxmsg;
    long int mq_msgsize;
    long int mq_curmsgs;
    long int __pad[4];
};
extern int mq_close(mqd_t);
extern int mq_getattr(mqd_t, struct mq_attr *);
extern int mq_notify(mqd_t, const struct sigevent *);
extern mqd_t mq_open(const char *, int, ...);
extern ssize_t mq_receive(mqd_t, char *, size_t, unsigned int *);
extern int mq_send(mqd_t, const char *, size_t, unsigned int);
extern int mq_setattr(mqd_t, const struct mq_attr *, struct mq_attr *);
extern ssize_t mq_timedreceive(mqd_t, char *, size_t, unsigned int *,
    const struct timespec *);
extern int mq_timedsend(mqd_t, const char *, size_t, unsigned int,
    const struct timespec *);
extern int mq_unlink(const char *);
13.19 Interfaces for libcrypt

Table 13-51 defines the library name and shared object name for the libcrypt library

Table 13-51 libcrypt Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libcrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libcrypt.so.1</td>
</tr>
</tbody>
</table>

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


13.19.1 Encryption

13.19.1.1 Interfaces for Encryption

An LSB conforming implementation shall provide the generic functions for Encryption specified in Table 13-52, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-52 libcrypt - Encryption Function Interfaces


13.20 Interfaces for libpam

Table 13-53 defines the library name and shared object name for the libpam library

Table 13-53 libpam Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libpam</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libpam.so.0</td>
</tr>
</tbody>
</table>

The Pluggable Authentication Module (PAM) interfaces allow applications to request authentication via a system administrator defined mechanism, known as a service.

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.

Note: Future versions of this specification might define additional service names.

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

[LSB] This Specification

13.20.1 Pluggable Authentication API

13.20.1.1 Interfaces for Pluggable Authentication API

An LSB conforming implementation shall provide the generic functions for Pluggable Authentication API specified in Table 13-54, with the full mandatory functionality as described in the referenced underlying specification.
Table 13-54 libpam - Pluggable Authentication API Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>pam_acct_mgmt</td>
<td>pam_authenticate</td>
<td>pam_chauthtok</td>
<td>pam_close_session</td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pam_end</td>
<td>pam_fail_delay</td>
<td>pam_get_item</td>
<td>pam_getenv</td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pam_getenvlist</td>
<td>pam_open_session</td>
<td>pam_putenv</td>
<td>pam_set_item</td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pam_setcred</td>
<td>pam_start</td>
<td>pam_strerror</td>
<td></td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td></td>
</tr>
</tbody>
</table>

13.21 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

13.21.1 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
#define PAM_SERVICE 1
#define PAM_USER 2
#define PAM_TTY 3
#define PAM_RHOST 4
#define PAM_CONV        5
#define PAM_RUSER       8
#define PAM_USER_PROMPT 9
#define PAM_SUCCESS     0
#define PAM_OPEN_ERR    1
#define PAM_USER_UNKNOWN 10
#define PAM_MAXTRIES   11
#define PAM_NEW_AUTHTOK_REQD 12
#define PAM_ACCT_EXPIRED 13
#define PAM_SESSION_ERR 14
#define PAM_CRED_UNAVAIL 15
#define PAM_CRED_EXPIRED 16
#define PAM_CRED_ERR    17
#define PAM_CONV_ERR    19
#define PAM_SYMBOL_ERR  2
#define PAM_AUTHTOK_ERR 20
#define PAM_AUTHTOK_RECOVER_ERR 21
#define PAM_AUTHTOK_LOCK_BUSY 22
#define PAM_TRY_AGAIN 24
#define PAM_ABORT       26
#define PAM_AUTHTOK_EXPIRED 27
#define PAM_BAD_ITEM    29
#define PAM_SERVICE_ERR 3
#define PAM_SYSTEM_ERR  4
#define PAM_BUF_ERR     5
#define PAM_PERM_DENIED 6
#define PAM_CREATION    7
#define PAM_CRED_INSUFFICIENT 8
#define PAM_AUTHINFO_UNAVAIL 9
#define PAM_DISALLOW_NULL_AUTHTOK 0x0001U
#define PAM_ESTABLISH_CRED 0x0002U
#define PAM_DELETE_CRED 0x0004U
#define PAM_REINITIALIZE_CRED 0x0008U
#define PAM_REFRESH_CRED 0x0010U
#define PAM_CHANGE_EXPIRED_AUTHTOK 0x0020U
#define PAM_SILENT      0x8000U

extern int pam_set_item(pam_handle_t *, int, const void *);
extern int pam_get_item(const pam_handle_t *, int, const void **);
extern const char *pam_strerror(pam_handle_t *, int);
extern char **pam_getenvlist(pam_handle_t *);
extern int pam_fail_delay(pam_handle_t *, unsigned int);
extern int pam_start(const char *, const char *, const struct pam_conv *,
  pam_handle_t * *);
extern int pam_end(pam_handle_t *, int);
extern int pam_authenticate(pam_handle_t *, int);
extern int pam_acct_mgmt(pam_handle_t *, int);
extern int pam_open_session(pam_handle_t *, int);
extern int pam_close_session(pam_handle_t *, int);
extern int pam_chauthtok(pam_handle_t *, int);
extern const char *pam_getenv(const pam_handle_t *, const char *);
extern int pam_putenv(const pam_handle_t *, const char *

13.22 Interface Definitions for libpam

The interfaces defined on the following pages are included in libpam and are
defined by this specification. Unless otherwise noted, these interfaces shall be
included in the source standard.
Other interfaces listed in Section 13.20 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**

```c
#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.

**Note:** Errors may be translated to text with `pam_strerror()`. 
pamAuthenticate

Name
pamAuthenticate — authenticate the user

Synopsis

#include <security/pam_appl.h>
int pam_authenticate(pam_handle_t * pamh, int flags);

Description
pamAuthenticate() 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.

Note: Errors may be translated to text with pam_strerror().
**pam_chauthtok**

**Name**

pam_chauthtok — change the authentication token for a given user

**Synopsis**

```c
#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_AUHTOK**

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.

**Note:** 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.

Note: 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.

Note: 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.

Note: Errors may be translated to text with pam_strerror().
pam_get_item

**Name**

pam_get_item — obtain the value of the indicated item.

**Synopsis**

```c
#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 overwritten.

**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.

  **Note:** Errors may be translated to text with `pam_strerror()`.

pam_getenv

**Name**

pam_getenv — get a PAM environment variable

**Synopsis**

```c
#include <security/pam_appl.h>
const char * pam_getenv(const pam_handle_t * pamh, const char * name);
```

**Description**

The `pam_getenv()` function shall search the environment associated with the PAM handle `pamh` for the environment variable `name`. If the specified environment variable cannot be found, a null pointer shall be returned. The application shall ensure that it does not modify the string pointed to by the `pam_getenv()` function.

**Return Value**

On success, `pam_getenv()` returns a pointer to a string of the form `name=value`. 
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 — indicate session has started

Synopsis
#include <security/pam_appl.h>
int pam_open_session(pam_handle_t * pamh, int flags);

Description
The pam_open_session() function is used to indicate that an authenticated
session has begun, after the user has been identified (see pam_authenticate())
and, if necessary, granted credentials (see pam_setcred()). 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 result 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.

Note: Errors may be translated to text with pam_strerror().
pam_putenv

Name

pam_putenv — Add, replace or delete a PAM environment variable

Synopsis

#include <security/pam_appl.h>
int pam_putenv(const pam_handle_t * pamh, const char * name_value);

Description

The pam_putenv() function shall modify the environment list associated with pamh. If name_value contains an '=' character, the characters to the left of the first '=' character represent the name, and the remaining characters after the '=' represent the value.

If the name environment variable exists in the environment associated with pamh, it shall be modified to have the value value. Otherwise, the name shall be added to the environment associated with pamh with the value value.

If there is no '=' character in name_value, the variable in the environment associated with pamh named name_value shall be deleted.

Return Value

On success, the pam_putenv() function shall return PAM_SUCCESS. Otherwise the return value indicates the error:

PAM_PERM_DENIED

The name_value argument is a null pointer.

PAM_BAD_ITEM

The PAM environment variable named name_value does not exist and therefore cannot be deleted.

PAM_ABORT

The PAM handle identified by pamh is corrupt.

PAM_BUF_ERR

Memory buffer error.
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.

**Note:** 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 may be 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.

Note: Errors may be translated to text with pam_strerror().
**pam_start**

**Name**

pam_start — initialize the PAM library

**Synopsis**

```c
#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,

```c
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**

```c
#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. Otherwise, a string indicating that the error is unknown shall be returned. It is unspecified whether or not the string returned is translated according to the setting of `LC_MESSAGES`.
IV Utility Libraries
14 Utility Libraries

14.1 Introduction

An LSB-conforming implementation shall also support the following 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.

- libz
- libcurses
- libutil

The structure of the definitions for these libraries follows the same model as used for Base Libraries.

14.2 Interfaces for libz

Table 14-1 defines the library name and shared object name for the libz library.

Table 14-1 libz Definition

<table>
<thead>
<tr>
<th>Library</th>
<th>libz</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME</td>
<td>libz.so.1</td>
</tr>
</tbody>
</table>

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

[LSB] This Specification

14.2.1 Compression Library

14.2.1.1 Interfaces for Compression Library

An LSB conforming implementation shall provide the generic functions for Compression Library specified in Table 14-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-2 libz - Compression Library Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>adler32 [LSB]</td>
<td>compress [LSB]</td>
<td>compress2 [LSB]</td>
<td>compressBound</td>
</tr>
<tr>
<td>crc32 [LSB]</td>
<td>deflate [LSB]</td>
<td>deflateBound</td>
<td>deflateCopy [LSB]</td>
</tr>
<tr>
<td>deflateEnd [LSB]</td>
<td>deflateInit2_</td>
<td>deflateInit_</td>
<td>deflateParams</td>
</tr>
<tr>
<td>deflateReset [LSB]</td>
<td>deflateSetDictionary</td>
<td>get_crc_table</td>
<td>gzclose [LSB]</td>
</tr>
<tr>
<td>gzseek [LSB]</td>
<td>gzsetparams</td>
<td>gztell [LSB]</td>
<td>gzwrite [LSB]</td>
</tr>
</tbody>
</table>
14.3 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

14.3.1 zlib.h

In addition to the values below, the zlib.h header shall define the ZLIB_VERSION macro. This macro may be used to check that the version of the library at run time matches that at compile time.

See also the zlibVersion() function, which returns the library version at run time. The first character of the version at compile time should always match the first character at run time.

```c
#define Z_NULL 0
#define MAX_WBITS 15
#define MAX_MEM_LEVEL 9
#define deflateInit2(stream, level, method, windowBits, memLevel, strategy)   
  deflateInit2_((stream), (level), (method), (windowBits), (memLevel), (strategy), ZLIB_VERSION, sizeof(z_stream))
#define deflateInit(stream, level)       
  deflateInit_((stream), (level), ZLIB_VERSION, sizeof(z_stream))
#define inflateInit2(stream, windowBits)   
  inflateInit2_((stream), (windowBits), ZLIB_VERSION, sizeof(z_stream))
#define inflateInit(stream)       
  inflateInit_((stream), ZLIB_VERSION, sizeof(z_stream))

typedef char charf;
typedef int intf;
```
typedef void *voidpf;
typedef unsigned int uInt;
typedef unsigned long int 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_BLOCK 5
#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_VERSION_ERROR (-6)
#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

extern int gzread(gzFile, voidp, unsigned int);
extern int gzcloses(gzFile);
extern gzFile gzopen(const char *, const char *);
extern gzFile gzdopen(int, const char *);
extern int gzwrites(gzFile, voidpc, unsigned int);
extern int gzfloses(gzFile, int);
extern const char *gzerror(gzFile, int *);
extern uLong adler32(uLong, const Bytef *, uInt);
extern int compress(Bytef *, uLongf *, const Bytef *, uLong);
extern int compress2(Bytef *, uLongf *, const Bytef *, uLong, int);
extern uLong crc32(uLong, const Bytef *, uInt);
extern int deflate(z_streamp, int);
extern int deflateCopy(z_streamp, z_streamp);
extern int deflateEnd(z_streamp);
extern int deflateInit2_(z_streamp, int, int, int, int, int);
extern int deflateInit_(z_streamp, int, const char *, int);
extern int deflateParams(z_streamp, int, int);
extern int deflateReset(z_streamp);
extern int deflateSetDictionary(z_streamp, const Bytef *, uInt);
extern const uLongf *get_crc_table(void);
extern int gzeof(gzFile);
extern int gzgetc(gzFile);
extern char *gzgets(gzFile, char *, int);
extern int gzprintf(gzFile, const char *, ...);
extern int gzputc(gzFile, int);
extern int gzputs(gzFile, const char *);
extern int gzrewind(gzFile);
extern z_off_t gzseek(gzFile, z_off_t, int);
extern int gzsetparams(gzFile, int, int);
extern const char *zError(int);
extern const char *zlibVersion(void);
extern uLong deflateBound(z_streamp, uLong);
extern uLong compressBound(uLong);

14.4 Interface Definitions for libz

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

Other interfaces listed in Section 14.2 shall behave as described in the referenced
base document.
adler32

Name
adler32 — compute Adler 32 Checksum

Synopsis
#include <zlib.h>
ulong adler32(ulong adler, const byte *buf, uint len);

Description
The adler32() function shall compute a running Adler-32 checksum (as
described in RFC 1950: ZLIB Compressed Data Format Specication). On entry,
adler is the previous value for the checksum, and buf shall point to an array of
len bytes of data to be added to this checksum. The adler32() function shall
return the new checksum.
If buf is NULL (or Z_NULL), adler32() shall return the initial checksum.

Return Value
The adler32() function shall return the new checksum value.

Errors
None defined.

Application Usage (informative)
The following code fragment demonstrates typical usage of the adler32() function:

    ulong adler = adler32(0L, Z_NULL, 0);
    while (read_buffer(buffer, length) != EOF) {
        adler = adler32(adler, buffer, length);
    }
    if (adler != original_adler) error();
compress

Name

compress — compress data

Synopsis

#include <zlib.h>
int compress(Bytef * dest, uLongf * destLen, const Bytef * source, uLong sourceLen);

Description

The compress() function shall attempt to compress sourceLen bytes of data in the buffer source, placing the result in the buffer dest.

On entry, destLen should point to a value describing the size of the dest buffer. The application should ensure that this value be at least (sourceLen × 1.001) + 12. On successful exit, the variable referenced by destLen shall be updated to hold the length of compressed data in dest.

The compress() function is equivalent to compress2() with a level of Z_DEFAULT_COMPRESSION.

Return Value

On success, compress() shall return Z_OK. Otherwise, compress() shall return a value to indicate the error.

Errors

On error, compress() shall return a value as described below:

Z_BUF_ERROR

The buffer dest was not large enough to hold the compressed data.

Z_MEM_ERROR

Insufficient memory.
compress2

Name
compress2 — compress data at a specified level

Synopsis
#include <zlib.h>
int compress2(Bytef * dest, uLongf * destLen, const Bytef * source, uLong sourceLen, int level);

Description
The compress2() function shall attempt to compress sourceLen bytes of data in the buffer source, placing the result in the buffer dest, at the level described by level. The level supplied shall be a value between 0 and 9, or the value Z_DEFAULT_COMPRESSION. A level of 1 requests the highest speed, while a level of 9 requests the highest compression. A level of 0 indicates that no compression should be used, and the output shall be the same as the input.

On entry, destLen should point to a value describing the size of the dest buffer. The application should ensure that this value be at least (sourceLen × 1.001) + 12. On successful exit, the variable referenced by destLen shall be updated to hold the length of compressed data in dest.

The compress() function is equivalent to compress2() with a level of Z_DEFAULT_COMPRESSION.

Return Value
On success, compress2() shall return Z_OK. Otherwise, compress2() shall return a value to indicate the error.

Errors
On error, compress2() shall return a value as described below:

- Z_BUF_ERROR
  The buffer dest was not large enough to hold the compressed data.

- Z_MEM_ERROR
  Insufficient memory.

- Z_STREAM_ERROR
  The level was not Z_DEFAULT_COMPRESSION, or was not between 0 and 9.
**compressBound**

**Name**

`compressBound` — compute compressed data size

**Synopsis**

```c
#include <zlib.h>
int compressBound(uLong sourceLen);
```

**Description**

The `compressBound()` function shall estimate the size of buffer required to compress `sourceLen` bytes of data using the `compress()` or `compress2()` functions. If successful, the value returned shall be an upper bound for the size of buffer required to compress `sourceLen` bytes of data, using the parameters stored in `stream`, in a single call to `compress()` or `compress2()`.

**Return Value**

The `compressBound()` shall return a value representing the upper bound of an array to allocate to hold the compressed data in a single call to `compress()` or `compress2()`. This function may return a conservative value that may be larger than `sourceLen`.

**Errors**

None defined.
**crc32**

**Name**

crc32 — compute CRC-32 Checksum

**Synopsis**

```c
#include <zlib.h>
uLong crc32(uLong crc, const Bytef *buf, uInt len);
```

**Description**

The `crc32()` function shall compute a running Cyclic Redundancy Check checksum, as defined in ITU-T V.42. On entry, `crc` is the previous value for the checksum, and `buf` shall point to an array of `len` bytes of data to be added to this checksum. The `crc32()` function shall return the new checksum.

If `buf` is `NULL` (or `Z_NULL`), `crc32()` shall return the initial checksum.

**Return Value**

The `crc32()` function shall return the new checksum value.

**Errors**

None defined.

**Application Usage (informative)**

The following code fragment demonstrates typical usage of the `crc32()` function:

```c
uLong crc = crc32(0L, Z_NULL, 0);
while (read_buffer(buffer, length) != EOF) {
    crc = crc32(crc, buffer, length);
} 
if (crc != original_crc) error();
```
deflate

Name
deflate — compress data

Synopsis
#include <zlib.h>
int deflate(z_stream *stream, int flush);

Description
The deflate() function shall attempt to compress data until either the input buffer is empty or the output buffer is full. The stream references a z_stream structure. Before the first call to deflate(), this structure should have been initialized by a call to deflateInit2().

Note: deflateInit2() is only in the binary standard; source level applications should initialize stream via a call to deflateInit() or deflateInit2().

In addition, the stream input and output buffers should have been initialized as follows:

next_in
should point to the data to be compressed.

avail_in
should contain the number of bytes of data in the buffer referenced by next_in.

next_out
should point to a buffer where compressed data may be placed.

avail_out
should contain the size in bytes of the buffer referenced by next_out.

The deflate() function shall perform one or both of the following actions:

1. Compress input data from next_in and update next_in, avail_in and total_in to reflect the data that has been compressed.

2. Fill the output buffer referenced by next_out, and update next_out, avail_out and total_out to reflect the compressed data that has been placed there. If flush is not Z_NO_FLUSH, and avail_out indicates that there is still space in output buffer, this action shall always occur (see below for further details).

The deflate() function shall return when either avail_in reaches zero (indicating that all the input data has been compressed), or avail_out reaches zero (indicating that the output buffer is full).

On success, the deflate() function shall set the adler field of the stream to the adler32() checksum of all the input data compressed so far (represented by total_in).
If the `deflate()` function shall attempt to determine the type of input data, and set field `data_type` in `stream` to `Z_ASCII` if the majority of the data bytes fall within the ASCII (ISO 646) printable character range. Otherwise, it shall set `data_type` to `Z_BINARY`. This data type is informational only, and does not affect the compression algorithm.

**Note:** Future versions of the LSB may remove this requirement, since it is based on an outdated character set that does not support Internationalization, and does not affect the algorithm. It is included for information only at this release. Applications should not depend on this field.

### Flush Operation

The parameter `flush` determines when compressed bits are added to the output buffer in `next_out`. If `flush` is `Z_NO_FLUSH`, `deflate()` may return with some data pending output, and not yet added to the output buffer.

If `flush` is `Z_SYNC_FLUSH`, `deflate()` shall flush all pending output to `next_out` and align the output to a byte boundary. A synchronization point is generated in the output.

If `flush` is `Z_FULL_FLUSH`, all output shall be flushed, as for `Z_SYNC_FLUSH`, and the compression state shall be reset. A synchronization point is generated in the output.

**Rationale:** `Z_SYNC_FLUSH` is intended to ensure that the compressed data contains all the data compressed so far, and allows a decompressor to reconstruct all of the input data. `Z_FULL_FLUSH` allows decompression to restart from this point if the previous compressed data has been lost or damaged. Flushing is likely to degrade the performance of the compression system, and should only be used where necessary.

If `flush` is set to `Z_FINISH`, all pending input shall be processed and `deflate()` shall return with `Z_STREAM_END` if there is sufficient space in the output buffer at `next_out`, as indicated by `avail_out`. If `deflate()` is called with `flush` set to `Z_FINISH` and there is insufficient space to store the compressed data, and no other error has occurred during compression, `deflate()` shall return `Z_OK`, and the application should call `deflate()` again with `flush` unchanged, and having updated `next_out` and `avail_out`.

If all the compression is to be done in a single step, `deflate()` may be called with `flush` set to `Z_FINISH` immediately after the stream has been initialized if `avail_out` is set to at least the value returned by `deflateBound()`.

### Return Value

On success, `deflate()` shall return `Z_OK`, unless `flush` was set to `Z_FINISH` and there was sufficient space in the output buffer to compress all of the input data. In this case, `deflate()` shall return `Z_STREAM_END`. On error, `deflate()` shall return a value to indicate the error.

**Note:** If `deflate()` returns `Z_OK` and has set `avail_out` to zero, the function should be called again with the same value for `flush`, and with updated `next_out` and `avail_out` until `deflate()` returns with `Z_OK` (or `Z_STREAM_END` if `flush` is set to `Z_FINISH`) and a non-zero `avail_out`.

### Errors

On error, `deflate()` shall return a value as described below, and set the `msg` field of `stream` to point to a string describing the error:
Z_BUF_ERROR

No progress is possible; either avail_in or avail_out was zero.

Z_MEM_ERROR

Insufficient memory.

Z_STREAM_ERROR

The state (as represented in stream) is inconsistent, or stream was NULL.

deflateBound

Name
deflateBound — compute compressed data size

Synopsis

#include <zlib.h>
int deflateBound(z_streamp stream, uLong sourceLen);

Description

The deflateBound() function shall estimate the size of buffer required to
compress sourceLen bytes of data. If successful, the value returned shall be an
upper bound for the size of buffer required to compress sourceLen bytes of
data, using the parameters stored in stream, in a single call to deflate() with
flush set to Z_FINISH.

On entry, stream should have been initialized via a call to deflateInit_() or
deflateInit2_().

Return Value

The deflateBound() shall return a value representing the upper bound of an
array to allocate to hold the compressed data in a single call to deflate(). If the
stream is not correctly initialized, or is NULL, then deflateBound() may return
a conservative value that may be larger than sourceLen.

Errors

None defined.
**deflateCopy**

**Name**

deflateCopy — copy compression stream

**Synopsis**

```c
#include <zlib.h>
int deflateCopy(z_streamp dest, z_streamp source);
```

**Description**

The `deflateCopy()` function shall copy the compression state information in `source` to the uninitialized `z_stream` structure referenced by `dest`.

On successful return, `dest` will be an exact copy of the stream referenced by `source`. The input and output buffer pointers in `next_in` and `next_out` will reference the same data.

**Return Value**

On success, `deflateCopy()` shall return `Z_OK`. Otherwise it shall return a value less than zero to indicate the error.

**Errors**

On error, `deflateCopy()` shall return a value as described below:

- **Z_STREAM_ERROR**
  
  The state in `source` is inconsistent, or either `source` or `dest` was NULL.

- **Z_MEM_ERROR**
  
  Insufficient memory available.

**Application Usage (informative)**

This function can be useful when several compression strategies will be tried, for example when there are several ways of pre-processing the input data with a filter. The streams that will be discarded should then be freed by calling `deflateEnd()`. Note that `deflateCopy()` duplicates the internal compression state which can be quite large, so this strategy may be slow and can consume lots of memory.
deflateEnd

Name
deflateEnd — free compression stream state

Synopsis
#include <zlib.h>
int deflateEnd(z_stream* stream);

Description
The deflateEnd() function shall free all allocated state information referenced by stream. All pending output is discarded, and unprocessed input is ignored.

Return Value
On success, deflateEnd() shall return Z_OK, or Z_DATA_ERROR if there was pending output discarded or input unprocessed. Otherwise it shall return Z_STREAM_ERROR to indicate the error.

Errors
On error, deflateEnd() shall return Z_STREAM_ERROR. The following conditions shall be treated as an error:
- The state in stream is inconsistent or inappropriate.
- stream is NULL.
deflateInit2_

**Name**

deflateInit2_ — initialize compression system

**Synopsis**

```c
#include <zlib.h>
int deflateInit2_ (z_streamp strm, int level, int method, int windowBits, int memLevel, int strategy, char * version, int stream_size);
```

**Description**

The `deflateInit2()` function shall initialize the compression system. On entry, `strm` shall refer to a user supplied `z_stream` object (a `z_stream_s` structure). The following fields shall be set on entry:

- `zalloc`:
  a pointer to an `alloc_func` function, used to allocate state information. If this is `NULL`, a default allocation function will be used.

- `zfree`:
  a pointer to a `free_func` function, used to free memory allocated by the `zalloc` function. If this is `NULL` a default free function will be used.

- `opaque`:
  If `alloc_func` is not `NULL`, `opaque` is a user supplied pointer to data that will be passed to the `alloc_func` and `free_func` functions.

If the `version` requested is not compatible with the version implemented, or if the size of the `z_stream_s` structure provided in `stream_size` does not match the size in the library implementation, `deflateInit2()` shall fail, and return `Z_VERSION_ERROR`.

The `level` supplied shall be a value between 0 and 9, or the value `Z_DEFAULT_COMPRESSION`. A `level` of 1 requests the highest speed, while a `level` of 9 requests the highest compression. A `level` of 0 indicates that no compression should be used, and the output shall be the same as the input.

The `method` selects the compression algorithm to use. LSB conforming implementation shall support the `Z_DEFLATED` method, and may support other implementation defined methods.

The `windowBits` parameter shall be a base 2 logarithm of the window size to use, and shall be a value between 8 and 15. A smaller value will use less memory, but will result in a poorer compression ratio, while a higher value will give better compression but utilize more memory.

The `memLevel` parameter specifies how much memory to use for the internal state. The value of `memLevel` shall be between 1 and `MAX_MEM_LEVEL`. Smaller values use less memory but are slower, while higher values use more memory to gain compression speed.

The `strategy` parameter selects the compression strategy to use:

`Z_DEFAULT_STRATEGY`
use the system default compression strategy. \texttt{Z\_DEFAULT\_STRATEGY} is particularly appropriate for text data.

\texttt{Z\_FILTERED}

use a compression strategy tuned for data consisting largely of small values with a fairly random distribution. \texttt{Z\_FILTERED} uses more Huffman encoding and less string matching than \texttt{Z\_DEFAULT\_STRATEGY}.

\texttt{Z\_HUFFMAN\_ONLY}

force Huffman encoding only, with no string match.

The \texttt{deflateInit2()} function is not in the source standard; it is only in the binary standard. Source applications should use the \texttt{deflateInit2()} macro.

**Return Value**

On success, the \texttt{deflateInit2()} function shall return \texttt{Z\_OK}. Otherwise, \texttt{deflateInit2()} shall return a value as described below to indicate the error.

**Errors**

On error, \texttt{deflateInit2()} shall return one of the following error indicators:

\texttt{Z\_STREAM\_ERROR}

Invalid parameter.

\texttt{Z\_MEM\_ERROR}

Insufficient memory available.

\texttt{Z\_VERSION\_ERROR}

The version requested is not compatible with the library version, or the \texttt{z\_stream} size differs from that used by the library.

In addition, the \texttt{msg} field of the \texttt{strm} may be set to an error message.
deflateInit_

Name
deflateInit_ — initialize compression system

Synopsis
#include <zlib.h>
int deflateInit_(z_stream * stream, int level, const char * version, int stream_size);

Description
The deflateInit_() function shall initialize the compression system. On entry, stream shall refer to a user supplied z_stream object (a z_stream_s structure). The following fields shall be set on entry:

zalloc
a pointer to an alloc_func function, used to allocate state information. If this is NULL, a default allocation function will be used.

zfree
a pointer to a free_func function, used to free memory allocated by the zalloc function. If this is NULL a default free function will be used.

opaque
If alloc_func is not NULL, opaque is a user supplied pointer to data that will be passed to the alloc_func and free_func functions.

If the version requested is not compatible with the version implemented, or if the size of the z_stream_s structure provided in stream_size does not match the size in the library implementation, deflateInit_() shall fail, and return Z_VERSION_ERROR.

The level supplied shall be a value between 0 and 9, or the value Z_DEFAULT_COMPRESSION. A level of 1 requests the highest speed, while a level of 9 requests the highest compression. A level of 0 indicates that no compression should be used, and the output shall be the same as the input.

The deflateInit_() function is not in the source standard; it is only in the binary standard. Source applications should use the deflateInit() macro.

The deflateInit_() function is equivalent to

deflateInit2_(stream, level, Z_DEFLATED, MAX_WBITS, MAX_MEM_LEVEL,
Z_DEFAULT_STRATEGY, version,
stream_size);}

Return Value
On success, the deflateInit_() function shall return Z_OK. Otherwise, deflateInit_() shall return a value as described below to indicate the error.

Errors
On error, deflateInit_() shall return one of the following error indicators:

Z_STREAM_ERROR
Invalid parameter.

Z_MEM_ERROR
Insufficient memory available.

Z_VERSION_ERROR
The version requested is not compatible with the library version, or the z_stream size differs from that used by the library.

In addition, the msg field of the stream may be set to an error message.
deflateParams

Name
deflateParams — set compression parameters

Synopsis
#include <zlib.h>
int deflateParams(z_streamp stream, int level, int strategy);

Description
The deflateParams() function shall dynamically alter the compression
parameters for the compression stream object stream. On entry, stream shall
refer to a user supplied z_stream object (a z_stream_s structure), already
initialized via a call to deflateInit_() or deflateInit2_().

The level supplied shall be a value between 0 and 9, or the value
Z_DEFAULT_COMPRESSION. A level of 1 requests the highest speed, while a
level of 9 requests the highest compression. A level of 0 indicates that no
compression should be used, and the output shall be the same as the input. If
the compression level is altered by deflateParams(), and some data has
already been compressed with this stream (i.e. total_in is not zero), and the
new level requires a different underlying compression method, then stream
shall be flushed by a call to deflate().

The strategy parameter selects the compression strategy to use:

Z_DEFAULT_STRATEGY
    use the system default compression strategy. Z_DEFAULT_STRATEGY is
    particularly appropriate for text data.

Z_FILTERED
    use a compression strategy tuned for data consisting largely of small values
    with a fairly random distribution. Z_FILTERED uses more Huffman
    encoding and less string matching than Z_DEFAULT_STRATEGY.

Z_HUFFMAN_ONLY
    force Huffman encoding only, with no string match.

Return Value
On success, the deflateParams() function shall return Z_OK. Otherwise,
deflateParams() shall return a value as described below to indicate the error.

Errors
On error, deflateParams() shall return one of the following error indicators:

Z_STREAM_ERROR
    Invalid parameter.

Z_MEM_ERROR
    Insufficient memory available.

Z_BUF_ERROR
    Insufficient space in stream to flush the current output.
In addition, the `msg` field of the `strm` may be set to an error message.

**Application Usage (Informative)**

Applications should ensure that the `stream` is flushed, e.g. by a call to `deflate(stream, Z_SYNC_FLUSH)` before calling `deflateParams()`, or ensure that there is sufficient space in `next_out` (as identified by `avail_out`) to ensure that all pending output and all uncompressed input can be flushed in a single call to `deflate()`.

**Rationale:** Although the `deflateParams()` function should flush pending output and compress all pending input, the result is unspecified if there is insufficient space in the output buffer. Applications should only call `deflateParams()` when the `stream` is effectively empty (flushed).

The `deflateParams()` can be used to switch between compression and straight copy of the input data, or to switch to a different kind of input data requiring a different strategy.

### deflateReset

**Name**

`deflateReset` — reset compression stream state

**Synopsis**

```c
#include <zlib.h>
int deflateReset(z_streamp stream);
```

**Description**

The `deflateReset()` function shall reset all state associated with `stream`. All pending output shall be discarded, and the counts of processed bytes (`total_in` and `total_out`) shall be reset to zero.

**Return Value**

On success, `deflateReset()` shall return `Z_OK`. Otherwise it shall return `Z_STREAM_ERROR` to indicate the error.

**Errors**

On error, `deflateReset()` shall return `Z_STREAM_ERROR`. The following conditions shall be treated as an error:

- The state in `stream` is inconsistent or inappropriate.
- `stream` is `NULL`.

© 2007 Linux Foundation
deflateSetDictionary

Name
deflateSetDictionary — initialize compression dictionary

Synopsis
#include <zlib.h>
int deflateSetDictionary(z_streamp stream, const Bytef * dictionary, uInt dictlen);

Description
The deflateSetDictionary() function shall initialize the compression dictionary associated with stream using the dictlen bytes referenced by dictionary.

The implementation may silently use a subset of the provided dictionary if the dictionary cannot fit in the current window associated with stream (see deflateInit2()). The application should ensure that the dictionary is sorted such that the most commonly used strings occur at the end of the dictionary.

If the dictionary is successfully set, the Adler32 checksum of the entire provided dictionary shall be stored in the adler member of stream. This value may be used by the decompression system to select the correct dictionary. The compression and decompression systems must use the same dictionary.

stream shall reference an initialized compression stream, with total_in zero (i.e. no data has been compressed since the stream was initialized).

Return Value
On success, deflateSetDictionary() shall return Z_OK. Otherwise it shall return Z_STREAM_ERROR to indicate an error.

Errors
On error, deflateSetDictionary() shall return a value as described below:

Z_STREAM_ERROR

The state in stream is inconsistent, or stream was NULL.

Application Usage (informative)
The application should provide a dictionary consisting of strings (ed note: do we really mean "strings"? Null terminated?) that are likely to be encountered in the data to be compressed. The application should ensure that the dictionary is sorted such that the most commonly used strings occur at the end of the dictionary.

The use of a dictionary is optional; however if the data to be compressed is relatively short and has a predictable structure, the use of a dictionary can substantially improve the compression ratio.
get_crc_table

Name
get_crc_table — generate a table for crc calculations

Synopsis
#include <zlib.h>
const uLongf * get_crc_table(void);

Description
Generate tables for a byte-wise 32-bit CRC calculation based on the polynomial:
\[ x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \]

In a multi-threaded application, get_crc_table() should be called by one thread to initialize the tables before any other thread calls any libz function.

Return Value
The get_crc_table() function shall return a pointer to the first of a set of tables used internally to calculate CRC-32 values (see crc32()).

Errors
None defined.
gzclose

Name

gzclose — close a compressed file stream

Synopsis

#include <zlib.h>
int gzclose (gzFile file );

Description

The gzclose() function shall close the compressed file stream file. If file
was open for writing, gzclose() shall first flush any pending output. Any state
information allocated shall be freed.

Return Value

On success, gzclose() shall return Z_OK. Otherwise, gzclose() shall return
an error value as described below.

Errors

On error, gzclose() may set the global variable errno to indicate the error. The
gzclose() shall return a value other than Z_OK on error.

Z_STREAM_ERROR

file was NULL (or Z_NULL), or did not refer to an open compressed file
stream.

Z_ERRNO

An error occurred in the underlying base libraries, and the application
should check errno for further information.

Z_BUF_ERROR

no compression progress is possible during buffer flush (see deflate()).
gzdopen

Name

gzdopen — open a compressed file

Synopsis

#include <zlib.h>
gzFile gzdopen ( int fd, const char * mode );

Description

The gzdopen() function shall attempt to associate the open file referenced by fd
with a gzFile object. The mode argument is based on that of fopen(), but the
mode parameter may also contain the following characters:

digit

set the compression level to digit. A low value (e.g. 1) means high speed,
while a high value (e.g. 9) means high compression. A compression level of
0 (zero) means no compression. See deflateInit2() for further details.

[fhR]

set the compression strategy to [fhR]. The letter f corresponds to filtered
data, the letter h corresponds to Huffman only compression, and the letter
R corresponds to Run Length Encoding. See deflateInit2() for further
details.

If fd refers to an uncompressed file, and mode refers to a read mode, gzdopen() shall
attempt to open the file and return a gzFile object suitable for reading
directly from the file without any decompression.

If mode is NULL, or if mode does not contain one of r, w, or a, gzdopen() shall
return Z_NULL, and need not set any other error condition.

Example

gzdopen(fileno(stdin), "r");

Attempt to associate the standard input with a gzFile object.

Return Value

On success, gzdopen() shall return a gzFile object. On failure, gzdopen() shall
return Z_NULL and may set errno accordingly.

Note: At version 1.2.2, zlib does not set errno for several error conditions.
Applications may not be able to determine the cause of an error.

Errors

On error, gzdopen() may set the global variable errno to indicate the error.
gzeof

Name

gzeof — check for end-of-file on a compressed file stream

Synopsis

#include <zlib.h>
int gzeof (gzFile file);

Description

The gzeof() function shall test the compressed file stream file for end of file.

Return Value

If file was open for reading and end of file has been reached, gzeof() shall
return 1. Otherwise, gzeof() shall return 0.

Errors

None defined.

gzerror

Name

gzerror — decode an error on a compressed file stream

Synopsis

#include <zlib.h>
const char * gzerror (gzFile file, int * errnum);

Description

The gzerror() function shall return a string describing the last error to have
occurred associated with the open compressed file stream referred to by file. It
shall also set the location referenced by errnum to an integer value that further
identifies the error.

Return Value

The gzerror() function shall return a string that describes the last error
associated with the given file compressed file stream. This string shall have the
format "%s: %s", with the name of the file, followed by a colon, a space,
and the description of the error. If the compressed file stream was opened by a
call to gzopen(), the format of the filename is unspecified.

Rationale: Although in all current implementations of libz file descriptors are
named "<fd:%d>", the code suggests that this is for debugging purposes only, and
may change in a future release.

It is unspecified if the string returned is determined by the setting of the
LC_MESSAGES category in the current locale.

Errors

None defined.
gzflush

Name

gzflush — flush a compressed file stream

Synopsis

#include <zlib.h>
int gzflush(gzFile file, int flush);

Description

The gzflush() function shall flush pending output to the compressed file stream identified by file, which must be open for writing.

Flush Operation

The parameter flush determines which compressed bits are added to the output file. If flush is Z_NO_FLUSH, gzflush() may return with some data pending output, and not yet written to the file.

If flush is Z_SYNC_FLUSH, gzflush() shall flush all pending output to file and align the output to a byte boundary. There may still be data pending compression that is not flushed.

If flush is Z_FULL_FLUSH, all output shall be flushed, as for Z_SYNC_FLUSH, and the compression state shall be reset. There may still be data pending compression that is not flushed.

Rationale: Z_SYNC_FLUSH is intended to ensure that the compressed data contains all the data compressed so far, and allows a decompressor to reconstruct all of the input data. Z_FULL_FLUSH allows decompression to restart from this point if the previous compressed data has been lost or damaged. Flushing is likely to degrade the performance of the compression system, and should only be used where necessary.

If flush is set to Z_FINISH, all pending uncompressed data shall be compressed and all output shall be flushed.

Return Value

On success, gzflush() shall return the value Z_OK. Otherwise gzflush() shall return a value to indicate the error, and may set the error number associated with the compressed file stream file.

Note: If flush is set to Z_FINISH and the flush operation is successful, gzflush() will return Z_OK, but the compressed file stream error value may be set to Z_STREAM_END.

Errors

On error, gzflush() shall return an error value, and may set the error number associated with the stream identified by file to indicate the error. Applications may use gzerror() to access this error value.

Z_ERRNO

An underlying base library function has indicated an error. The global variable errno may be examined for further information.

Z_STREAM_ERROR

© 2007 Linux Foundation
The stream is invalid, is not open for writing, or is in an invalid state.

Z_BUF_ERROR
   no compression progress is possible (see deflate()).

Z_MEM_ERROR
   Insufficient memory available to compress.

**gzgetc**

**Name**
gzgetc — read a character from a compressed file

**Synopsis**
```
#include <zlib.h>
int gzgetc (gzFile file);
```

**Description**
The `gzgetc()` function shall read the next single character from the compressed file stream referenced by `file`, which shall have been opened in a read mode (see `gzopen()` and `gzdopen()`).

**Return Value**
On success, `gzgetc()` shall return the uncompressed character read, otherwise, on end of file or error, `gzgetc()` shall return -1.

**Errors**
On end of file or error, `gzgetc()` shall return -1. Further information can be found by calling `gzerror()` with a pointer to the compressed file stream.
gzgets

Name

gzgets — read a string from a compressed file

Synopsis

```c
#include <zlib.h>
char * gzgets (gzFile file, char * buf, int len);
```

Description

The `gzgets()` function shall attempt to read data from the compressed file stream `file`, uncompressing it into `buf` until either `len-1` bytes have been inserted into `buf`, or until a newline character has been uncompressed into `buf`. A null byte shall be appended to the uncompressed data. The `file` shall have been opened in for reading (see `gzopen()` and `gzdopen()`).

Return Value

On success, `gzgets()` shall return a pointer to `buf`. Otherwise, `gzgets()` shall return Z_NULL. Applications may examine the cause using `gzerror()`.

Errors

On error, `gzgets()` shall return Z_NULL. The following conditions shall always be treated as an error:

- `file` is NULL, or does not refer to a file open for reading;
- `buf` is NULL;
- `len` is less than or equal to zero.
gzopen

Name
gzopen — open a compressed file

Synopsis
#include <zlib.h>
gzFile gzopen (const char *path, const char *mode);

Description
The gzopen() function shall open the compressed file named by path. The mode argument is based on that of fopen(), but the mode parameter may also contain the following characters:

digit
set the compression level to digit. A low value (e.g. 1) means high speed, while a high value (e.g. 9) means high compression. A compression level of 0 (zero) means no compression. See deflateInit2() for further details.

[fhR]
set the compression strategy to [fhR]. The letter f corresponds to filtered data, the letter h corresponds to Huffman only compression, and the letter R corresponds to Run Length Encoding. See deflateInit2() for further details.

If path refers to an uncompressed file, and mode refers to a read mode, gzopen() shall attempt to open the file and return a gzFile object suitable for reading directly from the file without any decompression.

If path or mode is NULL, or if mode does not contain one of r, w, or a, gzopen() shall return Z_NULL, and need not set any other error condition.

The gzFile object is also referred to as a compressed file stream.

Example
gzopen("file.gz", "w6h");

Attempt to create a new compressed file, file.gz, at compression level 6 using Huffman only compression.

Return Value
On success, gzopen() shall return a gzFile object (also known as a compressed file stream). On failure, gzopen() shall return Z_NULL and may set errno accordingly.

Note: At version 1.2.2, zlib does not set errno for several error conditions. Applications may not be able to determine the cause of an error.

Errors
On error, gzopen() may set the global variable errno to indicate the error.
gzprintf

Name
gzprintf — format data and compress

Synopsis
#include <zlib.h>
int gzprintf (gzFile file, const char * fmt, ...);

Description
The gzprintf() function shall format data as for fprintf(), and write the resulting string to the compressed file stream file.

Return Value
The gzprintf() function shall return the number of uncompressed bytes actually written, or a value less than or equal to 0 in the event of an error.

Errors
If file is NULL, or refers to a compressed file stream that has not been opened for writing, gzprintf() shall return Z_STREAM_ERROR. Otherwise, errors are as for gzwrite().

gzputc

Name
gzputc — write character to a compressed file

Synopsis
#include <zlib.h>
int gzputc (gzFile file, int c);

Description
The gzputc() function shall write the single character c, converted from integer to unsigned character, to the compressed file referenced by file, which shall have been opened in a write mode (see gzopen() and gzdopen()).

Return Value
On success, gzputc() shall return the value written, otherwise gzputc() shall return -1.

Errors
On error, gzputc() shall return -1.
gzputs

Name
gzputs — string write to a compressed file

Synopsis
#include <zlib.h>
int gzputs (gzFile file, const char * s);

Description
The gzputs() function shall write the null terminated string \textit{s} to the
compressed file referenced by \textit{file}, which shall have been opened in a write
mode (see gzopen() and gzdopen()). The terminating null character shall not
be written. The gzputs() function shall return the number of uncompressed
bytes actually written.

Return Value
On success, gzputs() shall return the number of uncompressed bytes actually
written to \textit{file}. On error gzputs() shall return a value less than or equal to 0.
Applications may examine the cause using gzerror().

Errors
On error, gzputs() shall set the error number associated with the stream
identified by \textit{file} to indicate the error. Applications should use gzerror() to
access this error value. If \textit{file} is NULL, gzputs() shall return Z_STREAM_ERR.

Z_ERRNO
An underlying base library function has indicated an error. The global
variable \texttt{errno} may be examined for further information.

Z_STREAM_ERROR
The stream is invalid, is not open for writing, or is in an invalid state.

Z_BUF_ERROR
no compression progress is possible (see deflate()).

Z_MEM_ERROR
Insufficient memory available to compress.
gzread

Name

gzread — read from a compressed file

Synopsis

#include <zlib.h>
int gzread (gzFile file, voidp buf, unsigned int len);

Description

The gzread() function shall read data from the compressed file referenced by file, which shall have been opened in a read mode (see gzopen() and gzdopen()). The gzread() function shall read data from file, and uncompress it into buf. At most, len bytes of uncompressed data shall be copied to buf. If the file is not compressed, gzread() shall simply copy data from file to buf without alteration.

Return Value

On success, gzread() shall return the number of bytes decompressed into buf. If gzread() returns 0, either the end-of-file has been reached or an underlying read error has occurred. Applications should use gzerror() or gzeof() to determine which occurred. On other errors, gzread() shall return a value less than 0 and applications may examine the cause using gzerror().

Errors

On error, gzread() shall set the error number associated with the stream identified by file to indicate the error. Applications should use gzerror() to access this error value.

Z_ERRNO

An underlying base library function has indicated an error. The global variable errno may be examined for further information.

Z_STREAM_END

End of file has been reached on input.

Z_DATA_ERROR

A CRC error occurred when reading data; the file is corrupt.

Z_STREAM_ERROR

The stream is invalid, or is in an invalid state.

Z_NEED_DICT

A dictionary is needed (see inflateSetDictionary()).

Z_MEM_ERROR

Insufficient memory available to decompress.
gzrewind

Name

gzrewind — reset the file-position indicator on a compressed file stream

Synopsis

#include <zlib.h>
int gzrewind(gzFile file);

Description

The gzrewind() function shall set the starting position for the next read on compressed file stream file to the beginning of file. file must be open for reading.
gzrewind() is equivalent to

(int)gzseek(file, 0L, SEEK_SET)

Return Value

On success, gzrewind() shall return 0. On error, gzrewind() shall return -1, and may set the error value for file accordingly.

Errors

On error, gzrewind() shall return -1, indicating that file is NULL, or does not represent an open compressed file stream, or represents a compressed file stream that is open for writing and is not currently at the beginning of file.
gzseek

Name
gzseek — reposition a file-position indicator in a compressed file stream

Synopsis

#include <zlib.h>

z_off_t gzseek(gzFile file, z_off_t offset, int whence);

Description

The gzseek() function shall set the file-position indicator for the compressed file stream file. The file-position indicator controls where the next read or write operation on the compressed file stream shall take place. The offset indicates a byte offset in the uncompressed data. The whence parameter may be one of:

SEEK_SET

the offset is relative to the start of the uncompressed data.

SEEK_CUR

the offset is relative to the current position in the uncompressed data.

Note: The value SEEK_END need not be supported.

If the file is open for writing, the new offset must be greater than or equal to the current offset. In this case, gzseek() shall compress a sequence of null bytes to fill the gap from the previous offset to the new offset.

Return Value

On success, gzseek() shall return the resulting offset in the file expressed as a byte position in the uncompressed data stream. On error, gzseek() shall return -1, and may set the error value for file accordingly.

Errors

On error, gzseek() shall return -1. The following conditions shall always result in an error:

• file is NULL
• file does not represent an open compressed file stream.
• file refers to a compressed file stream that is open for writing, and the newly computed offset is less than the current offset.
• The newly computed offset is less than zero.
• whence is not one of the supported values.

Application Usage (informative)

If file is open for reading, the implementation may still need to uncompress all of the data up to the new offset. As a result, gzseek() may be extremely slow in some circumstances.
gzsetparams

Name

gzsetparams — dynamically set compression parameters

Synopsis

#include <zlib.h>
int gzsetparams (gzFile file, int level, int strategy);

Description

The gzsetparams() function shall set the compression level and compression strategy on the compressed file stream referenced by file. The compressed file stream shall have been opened in a write mode. The level and strategy are as defined in deflateInit2. If there is any data pending writing, it shall be flushed before the parameters are updated.

Return Value

On success, the gzsetparams() function shall return Z_OK.

Errors

On error, gzsetparams() shall return one of the following error indications:

Z_STREAM_ERROR
Invalid parameter, or file not open for writing.

Z_BUF_ERROR
An internal inconsistency was detected while flushing the previous buffer.
gztell

Name
gztell — find position on a compressed file stream

Synopsis
#include <zlib.h>
z_off_t gztell (gzFile file);

Description
The gztell() function shall return the starting position for the next read or write operation on compressed file stream file. This position represents the number of bytes from the beginning of file in the uncompressed data.
gztell() is equivalent to
gzseek(file, 0L, SEEK_CUR)

Return Value
gztell() shall return the current offset in the file expressed as a byte position in the uncompressed data stream. On error, gztell() shall return -1, and may set the error value for file accordingly.

Errors
On error, gztell() shall return -1, indicating that file is NULL, or does not represent an open compressed file stream.
gzwrite

Name

gzwrite — write to a compressed file

Synopsis

#include <zlib.h>
int gzwrite (gzFile file, voidpc buf, unsigned int len);

Description

The gzwrite() function shall write data to the compressed file referenced by file, which shall have been opened in a write mode (see gzopen() and gzdopen()). On entry, buf shall point to a buffer containing len bytes of uncompressed data. The gzwrite() function shall compress this data and write it to file. The gzwrite() function shall return the number of uncompressed bytes actually written.

Return Value

On success, gzwrite() shall return the number of uncompressed bytes actually written to file. On error gzwrite() shall return a value less than or equal to 0. Applications may examine the cause using gzerror().

Errors

On error, gzwrite() shall set the error number associated with the stream identified by file to indicate the error. Applications should use gzerror() to access this error value.

Z_ERRNO

An underlying base library function has indicated an error. The global variable errno may be examined for further information.

Z_STREAM_ERROR

The stream is invalid, is not open for writing, or is in an invalid state.

Z_BUF_ERROR

no compression progress is possible (see deflate()).

Z_MEM_ERROR

Insufficient memory available to compress.
inflatable

Name

inflate — decompress data

Synopsis

#include <zlib.h>
int inflate(z_stream *stream, int flush);

Description

The inflate() function shall attempt to decompress data until either the input
buffer is empty or the output buffer is full. The stream references a z_stream
structure. Before the first call to inflate(), this structure should have been
initialized by a call to inflateInit2().

Note: inflateInit2() is only in the binary standard; source level applications
should initialize stream via a call to inflateInit() or inflateInit2().

In addition, the stream input and output buffers should have been initialized as
follows:

next_in

should point to the data to be decompressed.

avail_in

should contain the number of bytes of data in the buffer referenced by
next_in.

next_out

should point to a buffer where decompressed data may be placed.

avail_out

should contain the size in bytes of the buffer referenced by next_out.

The inflate() function shall perform one or both of the following actions:

1. Decompress input data from next_in and update next_in, avail_in and
total_in to reflect the data that has been decompressed.

2. Fill the output buffer referenced by next_out, and update next_out,
avail_out, and total_out to reflect the decompressed data that has been
placed there. If flush is not Z_NO_FLUSH, and avail_out indicates that
there is still space in output buffer, this action shall always occur (see
below for further details).

The inflate() function shall return when either avail_in reaches zero
(indicating that all the input data has been compressed), or avail_out reaches
zero (indicating that the output buffer is full).

Flush Operation

The parameter flush determines when uncompressed bytes are added to the
output buffer in next_out. If flush is Z_NO_FLUSH, inflate() may return with
some data pending output, and not yet added to the output buffer.

If flush is Z_SYNC_FLUSH, inflate() shall flush all pending output to
next_out, and update next_out and avail_out accordingly.
If `flush` is set to `Z_BLOCK`, `inflate()` shall stop adding data to the output buffer if and when the next compressed block boundary is reached (see RFC 1951: DEFLATE Compressed Data Format Specification).

If `flush` is set to `Z_FINISH`, all of the compressed input shall be decompressed and added to the output. If there is insufficient output space (i.e. the compressed input data uncompresses to more than `avail_out` bytes), then `inflate()` shall fail and return `Z_BUF_ERROR`.

**Return Value**

On success, `inflate()` shall return `Z_OK` if decompression progress has been made, or `Z_STREAM_END` if all of the input data has been decompressed and there was sufficient space in the output buffer to store the uncompressed result.

On error, `inflate()` shall return a value to indicate the error.

Note: If `inflate()` returns `Z_OK` and has set `avail_out` to zero, the function should be called again with the same value for `flush`, and with updated `next_out` and `avail_out` until `inflate()` returns with either `Z_OK` or `Z_STREAM_END` and a non-zero `avail_out`.

On success, `inflate()` shall set the `adler` to the Adler-32 checksum of the output produced so far (i.e. `total_out` bytes).

**Errors**

On error, `inflate()` shall return a value as described below, and may set the `msg` field of `stream` to point to a string describing the error:

- `Z_BUF_ERROR`
  
  No progress is possible; either `avail_in` or `avail_out` was zero.

- `Z_MEM_ERROR`
  
  Insufficient memory.

- `Z_STREAM_ERROR`
  
  The state (as represented in `stream`) is inconsistent, or `stream` was NULL.

- `Z_NEED_DICT`
  
  A preset dictionary is required. The `adler` field shall be set to the Adler-32 checksum of the dictionary chosen by the compressor.
inflaterEnd

Name

inflateEnd — free decompression stream state

Synopsis

#include <zlib.h>
int inflateEnd(z_streamp stream);

Description

The inflateEnd() function shall free all allocated state information referenced by stream. All pending output is discarded, and unprocessed input is ignored.

Return Value

On success, inflateEnd() shall return Z_OK. Otherwise it shall return Z_STREAM_ERROR to indicate the error.

Errors

On error, inflateEnd() shall return Z_STREAM_ERROR. The following conditions shall be treated as an error:

• The state in stream is inconsistent.
• stream is NULL.
• The zfree function pointer is NULL.
inflaterInit2_

Name

inflateInit2_ — initialize decompression system

Synopsis

#include <zlib.h>
int inflateInit2_ (z_streamp strm, int windowBits, char * version,
int stream_size);

Description

The inflateInit2_() function shall initialize the decompression system. On
entry, strm shall refer to a user supplied z_stream object (a z_stream_s
structure). The following fields shall be set on entry:

zalloc

a pointer to an alloc_func function, used to allocate state information. If this
is NULL, a default allocation function will be used.

zfree

a pointer to a free_func function, used to free memory allocated by the
zalloc function. If this is NULL a default free function will be used.

opaque

If alloc_func is not NULL, opaque is a user supplied pointer to data that
will be passed to the alloc_func and free_func functions.

If the version requested is not compatible with the version implemented, or if
the size of the z_stream_s structure provided in stream_size does not match
the size in the library implementation, inflateInit2_() shall fail, and return
Z_VERSION_ERROR.

The windowBits parameter shall be a base 2 logarithm of the maximum
window size to use, and shall be a value between 8 and 15. If the input data was
compressed with a larger window size, subsequent attempts to decompress this
data will fail with Z_DATA_ERROR, rather than try to allocate a larger window.

The inflateInit2_() function is not in the source standard; it is only in the
binary standard. Source applications should use the inflateInit2() macro.

Return Value

On success, the inflateInit2_() function shall return Z_OK. Otherwise,
inflateInit2_() shall return a value as described below to indicate the error.

Errors

On error, inflateInit2_() shall return one of the following error indicators:

Z_STREAM_ERROR

Invalid parameter.

Z_MEM_ERROR

Insufficient memory available.

Z_VERSION_ERROR
The version requested is not compatible with the library version, or the z_stream size differs from that used by the library.

In addition, the msg field of the strm may be set to an error message.
inflateInit_

Name
inflateInit_ — initialize decompression system

Synopsis

```
#include <zlib.h>
int inflateInit_(z_streamp stream, const char * version, int stream_size);
```

Description

The inflateInit_() function shall initialize the decompression system. On entry, stream shall refer to a user supplied z_stream object (a z_stream_s structure). The following fields shall be set on entry:

- `zalloc`
  - a pointer to an alloc_func function, used to allocate state information. If this is NULL, a default allocation function will be used.

- `zfree`
  - a pointer to a free_func function, used to free memory allocated by the `zalloc` function. If this is NULL a default free function will be used.

- `opaque`
  - If alloc_func is not NULL, opaque is a user supplied pointer to data that will be passed to the alloc_func and free_func functions.

If the version requested is not compatible with the version implemented, or if the size of the z_stream_s structure provided in stream_size does not match the size in the library implementation, inflateInit_() shall fail, and return Z_VERSION_ERROR.

The inflateInit_() function is not in the source standard; it is only in the binary standard. Source applications should use the inflateInit() macro.

The inflateInit_() shall be equivalent to

```
inflateInit2_(strm, MAX_WBITS, version, stream_size);
```

Return Value

On success, the inflateInit_() function shall return Z_OK. Otherwise, inflateInit_() shall return a value as described below to indicate the error.

Errors

On error, inflateInit_() shall return one of the following error indicators:

- Z_STREAM_ERROR
  - Invalid parameter.

- Z_MEM_ERROR
  - Insufficient memory available.

- Z_VERSION_ERROR
The version requested is not compatible with the library version, or the
z_stream size differs from that used by the library.

In addition, the msg field of the strm may be set to an error message.

inflateReset

Name

inflateReset — reset decompression stream state

Synopsis

#include <zlib.h>
int inflateReset(z_streamp stream);

Description

The inflateReset() function shall reset all state associated with stream. All
pending output shall be discarded, and the counts of processed bytes (total_in
and total_out) shall be reset to zero.

Return Value

On success, inflateReset() shall return Z_OK. Otherwise it shall return
Z_STREAM_ERROR to indicate the error.

Errors

On error, inflateReset() shall return Z_STREAM_ERROR. The following
conditions shall be treated as an error:

• The state in stream is inconsistent or inappropriate.
• stream is NULL.
inflateSetDictionary

**Name**
inflateSetDictionary — initialize decompression dictionary

**Synopsis**

```c
#include <zlib.h>
int inflateSetDictionary(z_streamp stream, const Bytef *dictionary, uInt dictlen);
```

**Description**
The `inflateSetDictionary()` function shall initialize the decompression dictionary associated with `stream` using the `dictlen` bytes referenced by `dictionary`.

The `inflateSetDictionary()` function should be called immediately after a call to `inflate()` has failed with return value Z_NEED_DICT. The `dictionary` must have the same Adler-32 checksum as the dictionary used for the compression (see `deflateSetDictionary()`).

`stream` shall reference an initialized decompression stream, with `total_in` zero (i.e. no data has been decompressed since the stream was initialized).

**Return Value**
On success, `inflateSetDictionary()` shall return Z_OK. Otherwise it shall return a value as indicated below.

**Errors**
On error, `inflateSetDictionary()` shall return a value as described below:

`Z_STREAM_ERROR`

The state in `stream` is inconsistent, or `stream` was NULL.

`Z_DATA_ERROR`

The Adler-32 checksum of the supplied dictionary does not match that used for the compression.

**Application Usage (informative)**
The application should provide a dictionary consisting of strings (ed note: do we really mean "strings"? Null terminated?) that are likely to be encountered in the data to be compressed. The application should ensure that the dictionary is sorted such that the most commonly used strings occur at the end of the dictionary.

The use of a dictionary is optional; however if the data to be compressed is relatively short and has a predictable structure, the use of a dictionary can substantially improve the compression ratio.
inflateSync

Name

inflateSync — advance compression stream to next sync point

Synopsis

#include <zlib.h>
int inflateSync(z_streamp stream);

Description

The `inflateSync()` function shall advance through the compressed data in 
stream, skipping any invalid compressed data, until the next full flush point is 
reached, or all input is exhausted. See the description for `deflate()` with flush 
level Z_FULL_FLUSH. No output is placed in `next_out`.

Return Value

On success, `inflateSync()` shall return Z_OK, and update the `next_in`, 
`avail_in`, and `total_in` fields of `stream` to reflect the number of bytes of 
compressed data that have been skipped. Otherwise, `inflateSync()` shall 
return a value as described below to indicate the error.

Errors

On error, `inflateSync()` shall return a value as described below:

Z_STREAM_ERROR

The state (as represented in `stream`) is inconsistent, or `stream` was NULL.

Z_BUF_ERROR

There is no data available to skip over.

Z_DATA_ERROR

No sync point was found.
inflateSyncPoint

**Name**

inflateSyncPoint — test for synchronization point

**Synopsis**

```c
#include <zlib.h>
int inflateSyncPoint(z_streamp stream);
```

**Description**

The `inflateSyncPoint()` function shall return a non-zero value if the compressed data stream referenced by `stream` is at a synchronization point.

**Return Value**

If the compressed data in `stream` is at a synchronization point (see `deflate()` with a flush level of `Z_SYNC_FLUSH` or `Z_FULL_FLUSH`), `inflateSyncPoint()` shall return a non-zero value, other than `Z_STREAM_ERROR`. Otherwise, if the `stream` is valid, `inflateSyncPoint()` shall return 0. If `stream` is invalid, or in an invalid state, `inflateSyncPoint()` shall return `Z_STREAM_ERROR` to indicate the error.

**Errors**

On error, `inflateSyncPoint()` shall return a value as described below:

- **Z_STREAM_ERROR**

  The state (as represented in `stream`) is inconsistent, or `stream` was NULL.
uncompress

Name
uncompress — uncompress data

Synopsis
#include <zlib.h>
int uncompress(Bytef * dest, uLongf * destLen, const Bytef * source,
uLong sourceLen);

Description
The uncompress() function shall attempt to uncompress sourceLen bytes of
data in the buffer source, placing the result in the buffer dest.

On entry, destLen should point to a value describing the size of the dest
buffer. The application should ensure that this value is large enough to hold the
entire uncompressed data.

Note: The LSB does not describe any mechanism by which a compressor can
communicate the size required to the uncompressor.

On successful exit, the variable referenced by destLen shall be updated to hold
the length of uncompressed data in dest.

Return Value
On success, uncompress() shall return Z_OK. Otherwise, uncompress() shall
return a value to indicate the error.

Errors
On error, uncompress() shall return a value as described below:

Z_BUF_ERROR
The buffer dest was not large enough to hold the uncompressed data.

Z_MEM_ERROR
Insufficient memory.

Z_DATA_ERROR
The compressed data (referenced by source) was corrupted.
zError

Name
zError — translate error number to string

Synopsis
#include <zlib.h>
const char * zError(int err);

Description
The zError() function shall return the string identifying the error associated with err. This allows for conversion from error code to string for functions such as compress() and uncompress(), that do not always set the string version of an error.

Return Value
The zError() function shall return a the string identifying the error associated with err, or NULL if err is not a valid error code.

It is unspecified if the string returned is determined by the setting of the LC_MESSAGES category in the current locale.

Errors
None defined.

zlibVersion

Name
zlibVersion — discover library version at run time

Synopsis
#include <zlib.h>
const char * zlibVersion(void);

Description
The zlibVersion() function shall return the string identifying the interface version at the time the library was built.

Applications should compare the value returned from zlibVersion() with the macro constant ZLIB_VERSION for compatibility.

Return Value
The zlibVersion() function shall return a the string identifying the version of the library currently implemented.

Errors
None defined.

14.5 Interfaces for libnncurses

Table 14-3 defines the library name and shared object name for the libnncurses library
Table 14-3 libncurses Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libncurses</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libncurses.so.5</td>
</tr>
</tbody>
</table>

The Parameters or return value of the following interface have had the const qualifier added as shown here.

```c
extern const char *keyname (int);
extern int mvscanw (int, int, const char *, ...);
extern int mvwscanw (WINDOW *, int, int, const char *, ...);
extern SCREEN *newterm (const char *, FILE *, FILE *);
extern int scanw (const char *, ...);
extern int vwscanw (WINDOW *, const char *, va_list);
extern int vw_scanw (WINDOW *, const char *, va_list);
extern int wscanw (WINDOW *, const char *, ...);
```

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

[SUS-CURSES] X/Open Curses

### 14.5.1 Curses

#### 14.5.1.1 Interfaces for Curses

An LSB conforming implementation shall provide the generic functions for Curses specified in Table 14-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-4 libncurses - Curses Function Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>clear [SUS-CURSES]</td>
<td>clearok [SUS-CURSES]</td>
<td>cwrtobot [SUS-CURSES]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>getwin [SUS-CURSES]</td>
<td>halfdelay [SUS-CURSES]</td>
<td>has_colors [SUS-CURSES]</td>
<td>has_ic [SUS-CURSES]</td>
</tr>
<tr>
<td>instr [SUS-CURSES]</td>
<td>intrflush [SUS-CURSES]</td>
<td>is_linetouched</td>
<td>is_wintouched</td>
</tr>
<tr>
<td>move [SUS-CURSES]</td>
<td>mvaddch [SUS-CURSES]</td>
<td>mvaddchnstr</td>
<td>mvaddchstr</td>
</tr>
<tr>
<td>mvline [SUS-CURSES]</td>
<td>mvwaddch [SUS-CURSES]</td>
<td>mvwaddchnstr</td>
<td>mvwaddchstr</td>
</tr>
<tr>
<td>mvwin [SUS-CURSES]</td>
<td>mvwinch [SUS-CURSES]</td>
<td>mvwinchnstr</td>
<td>mvwincstr [SUS-CURSES]</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic data interfaces for Curses specified in Table 14-5, with the full mandatory functionality as described in the referenced underlying specification.

**Table 14-5 libncurses - Curses Data Interfaces**

|----------------------|--------------------------|-------------------|--------------------|
14.6 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. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) 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.

14.6.1 curses.h

```c
#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'])
#define ACS_PLMINUS     (acs_map['g'])
#define ACS_BOARD       (acs_map['h'])
#define ACS_LANTERN     (acs_map['i'])
#define ACS_LRCORNER    (acs_map['j'])
#define ACS_URCORNER    (acs_map['k'])
#define ACS_ULCORNER    (acs_map['l'])
#define ACS_LLCORNER    (acs_map['m'])
#define ACS_PLUS        (acs_map['n'])
#define ACS_S1  (acs_map['o'])
#define ACS_HLINE       (acs_map['q'])
#define ACS_S9  (acs_map['s'])
#define ACS_LTEE        (acs_map['t'])
#define ACS_RTEE        (acs_map['u'])
#define ACS_BTEE        (acs_map['v'])
#define ACS_TTEE        (acs_map['w'])
#define ACS_VLINE       (acs_map['x'])
#define ACS_DIAMOND     (acs_map['`'])
#define ACS_BULLET      (acs_map['~'])

#define getmaxyx(win,y,x)       
  (y=(win)?((win)->_maxy+1):ERR,x=(win)?((win)->_maxx+1):ERR)
#define getbegyx(win,y,x)       
  (y=(win)?(win)->_begy:ERR,x=(win)?(win)->_begx:ERR)
#define getyx(win,y,x)  
  (y=(win)?(win)->_cury:ERR,x=(win)?(win)->_curx:ERR)
#define getparyx(win,y,x)       
  (y=(win)?(win)->_pary:ERR,x=(win)?(win)->_parx:ERR)

#define WA_ALTCHARSET   A_ALTCHARSET
```
#define WA_ATTRIBUTES   A_ATTRIBUTES
#define WA_BLINK        A_BLINK
#define WA_BOLD A_BOLD
#define WA_DIM  A_DIM
#define WA_HORIZONTAL   A_HORIZONTAL
#define WA_INVIS        A_INVIS
#define WA_LEFT A_LEFT
#define WA_LOW  A_LOW
#define WA_NORMAL       A_NORMAL
#define WA_PROTECT      A_PROTECT
#define WA_RIGHT        A_RIGHT
#define WA_STANDOUT     A_STANDOUT
#define WA_TOP  A_TOP
#define WA_UNDERLINE    A_UNDERLINE
#define WA_VERTICAL     A_VERTICAL
#define A_REVERSE       NCURSES_BITS(1UL,10)

#define COLOR_BLACK     0
#define COLOR_RED       1
#define COLOR_GREEN     2
#define COLOR_YELLOW    3
#define COLOR_BLUE      4
#define COLOR_MAGENTA   5
#define COLOR_CYAN      6
#define COLOR_WHITE     7

#define _SUBWIN 0x01
#define _ENDLINE 0x02
#define _FULLWIN 0x04
#define _SCROLLWIN 0x08
#define _ISPAD  0x10
#define _HASMOVED 0x20

typedef unsigned char bool;
typedef unsigned long int chtype;
typedef struct screen SCREEN;
typedef struct _win_st WINDOW;
typedef chtype attr_t;
typedef struct {
    attr_t attr;
    wchar_t chars[5];
} cchar_t;

struct pdat {
    short _pad_y;
    short _pad_x;
    short _pad_top;
    short _pad_left;
    short _pad_bottom;
    short _pad_right;
};

struct _win_st {
    short _cury;
    short _curx;
    short _maxy;
    short _maxx;
    short _begy;
    short _begx;
    short _flags;
    attr_t _attrs;
    chtype _bkgd;
    bool _notimeout;
    bool _clear;
    bool _leaveok;
bool _scroll;
bool _idlok;
bool _idcok;
bool _immed;
bool _sync;
bool _use_keypad;
int _delay;
struct ldat * _line;
short _regtop;
short _regbottom;
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
#define KEY_MARK 0554
#define KEY_MESSAGE 0555

#define KEY_MOVE        0556
#define KEY_NEXT        0557
#define KEY_OPEN        0560
#define KEY_OPTIONS     0561
#define KEY_PREVIOUS    0562
#define KEY_REDO        0563
#define KEY_REFERENCE   0564
#define KEY_REFRESH     0565
#define KEY_REPLACE     0566
#define KEY_RESTART     0567
#define KEY_RESUME      0570
#define KEY_SAVE        0571
#define KEY_SBEG        0572
#define KEY_SCANCEL     0573
#define KEY_SCOMMAND    0574
#define KEY_SCOPY       0575
#define KEY_SCREATE     0576
#define KEY_SDC 0577
#define KEY_SD 0600
#define KEY_SELECT      0601
#define KEY_SEND        0602
#define KEY_SEOL        0603
#define KEY_SEXIT       0604
#define KEY_SFIND       0605
#define KEY_SHOME       0606
#define KEY_SIC 0607
#define KEY_SLEFT       0608
#define KEY_SMES        0609
#define KEY_SMOV        0610
#define KEY_SNEX        0611
#define KEY_SOPT        0612
#define KEY_SPRV        0613
#define KEY_SPRINT      0614
#define KEY_SRDO        0615
#define KEY_SRPL        0616
#define KEY_SRSH        0617
#define KEY_ST     0618
#define KEY_SYM   0619
#define KEY_SZZ     0620
#define KEY_SZU     0621
#define KEY_SZV     0622
#define KEY_SZW     0623
#define KEY_SS 0624
#define KEY_SUSP 0625
#define KEY_SUNDO 0626
#define KEY_SUSPND 0627
#define KEY_SUND 0628
#define KEY_SUNDO 0629
#define KEY_SUNDO 0630
#define KEY_SUNDO 0631
#define KEY_SUSP 0632
#define KEY_MAX 0777
#define PAIR_NUMBER(a)  (((a)&A_COLOR)>>8)
#define NCURSES_BITS(mask,shift)        ((mask)<<((shift)+8))
#define A_CHARTEXT      (NCURSES_BITS(1UL,0)-1UL)
#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_HORIZNCURSES_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)

extern int addch(const chtype);
extern int addchnstr(const chtype *, int);
extern int addnstr(const char *, int);
extern int addstr(const char *);
extern int attroff(int);
extern int atron(int);
extern int attrset(int);
extern int attr_get(attr_t *, short *, void *);
extern int attr_off(attr_t, void *);
extern int attr_set(attr_t, short, void *);
extern int baudrate(void);
extern int beep(void);
extern int bkgd(chtype);
extern void bkgdset(chtype);
extern int border(chtype, chtype, chtype, chtype, chtype, chtype,
    chtype);
extern int box(WINDOW *, chtype, chtype);
extern bool can_change_color(void);
extern int cbreak(void);
extern int chgat(int, attr_t, short, const void *);
extern int clear(void);
extern int clearok(WINDOW *, bool);
extern int clrtobot(void);
extern int clrtoeol(void);
extern int color_content(short, short *, short *, short *);
extern int color_set(short, void *);
extern int copywin(const WINDOW *, WINDOW *, int, int, int, int,
    int, int, int);
extern int curs_set(int);
extern int def_prog_mode(void);
extern int def_shell_mode(void);
extern int delay_output(int);
extern int delch(void);
extern void delscreen(SCREEN *);
extern int delwin(WINDOW *);
extern int deleteln(void);
extern WINDOW *derwin(WINDOW *, int, int, int, int);
extern int doupdate(void);
extern WINDOW *dupwin(WINDOW *);
extern int echo(void);
extern int echochar(const chtype);
extern int erase(void);
extern int endwin(void);
extern char erasechar(void);
extern void filter(void);
extern int flash(void);
extern int flushinp(void);
extern chtype getbkgd(WINDOW *);
extern int getch(void);
extern int getnstr(char *, int);
extern int getstr(char *);
extern WINDOW *getwin(FILE *);
extern int halfdelay(int);
extern bool has_colors(void);
extern bool has_ic(void);
extern bool has_il(void);
extern int hline(chtype, int);
extern void idcok(WINDOW *, bool);
extern int idlok(WINDOW *, bool);
extern void immedok(WINDOW *, bool);
extern chtype inch(void);
extern int inchnstr(chtype *, int);
extern int inchstr(chtype *);
extern WINDOW *initscr(void);
extern int init_color(short, short, short, short);
extern int init_pair(short, short, short);
extern int innstr(char *, int);
extern int insdelln(int);
extern int insertln(void);
extern int insnstr(const char *, int);
extern int insstr(const char *);
extern int instr(char *);
extern int intrflush(WINDOW *, bool);
extern bool isendwin(void);
extern bool is_linetouched(WINDOW *, int);
extern bool is_wintouched(WINDOW *);
extern const char *keyname(int);
extern int keypad(WINDOW *, bool);
extern char killchar(void);
extern int leaveok(WINDOW *, bool);
extern char *longname(void);
extern int meta(WINDOW *, bool);
extern int move(int, int);
extern int mvaddch(int, int, const chtype);
extern int mvaddchnstr(int, int, const chtype *, int);
extern int mvaddchstr(int, int, const chtype *);
extern int mvaddlnstr(int, int, const char *, int);
extern int mvaddstr(int, int, const char *);
extern int mvchgat(int, int, int, attr_t, short, const void *);
extern int mvcur(int, int, int, int);
extern int mvdelch(int, int);
extern int mvderwin(WINDOW *, int, int);
extern int mvgetch(int, int);
extern int mvgetnstr(int, int, char *, int);
extern int mvgetstr(int, int, char *);
extern chtype mvhline(int, int, chtype, int);
extern int mvhline(char *, int, chtype *);
extern int mvinnstr(int, int, char *, int);
extern int mvinsch(int, int, chtype);
extern int mvinsnstr(int, int, const char *, int);
extern int mvinsstr(int, int, const char *);
extern int mvinsxstr(int, int, char *);
extern int mvprintw(int, int, char *, ...);
extern int mvscanw(int, int, const char *, ...);
extern int mvslnt(int, int, chtype, int);
extern int mvwaddch(WINDOW *, int, int, const chtype);
extern int mvwaddchnstr(WINDOW *, int, int, const chtype *, int);
extern int mvwaddchstr(WINDOW *, int, int, const chtype *);
extern int mvwaddlnstr(WINDOW *, int, int, const char *);
extern int mvwaddstr(WINDOW *, int, int, const char *);
extern int mvwaddxstr(WINDOW *, int, int, const char *);
extern int mvwchgat(WINDOW *, int, int, attr_t, short, const void *);
extern int mvwcur(int, int, int, int);
extern int mvwdelch(WINDOW *, int, int);
extern int mvwgetch(WINDOW *, int, int);
extern int mvwgetnstr(WINDOW *, int, int, char *, int);
extern int mvwgetstr(WINDOW *, int, int, char *);
extern chtype mvwline(WINDOW *, int, int, chtype, int);
extern int mvwin(WINDOW *, int, int);
extern chtype mvwinch(WINDOW *, int, int);
extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
extern int mvwinchstr(WINDOW *, int, int, chtype *);
extern int mvwinnstr(WINDOW *, int, int, char *, int);
extern int mvwinsch(WINDOW *, int, int, chtype);
extern int mvwinsnstr(WINDOW *, int, int, const char *, int);
extern int mvwinsstr(WINDOW *, int, int, const char *);
extern int mvwprintw(WINDOW *, int, int, char *, ...);
extern int mvwscanf(WINDOW *, int, int, const char *, ...);
extern int mvwvline(WINDOW *, int, int, chtype, int);
extern int napms(int);
extern WINDOW *newpad(int, int);
extern SCREEN *newterm(const char *, FILE *, FILE *);
extern WINDOW *newwin(int, int, int, int);
extern int nl(void);
extern int nocbreak(void);
extern int nodelay(WINDOW *, bool);
extern int noecho(void);
extern int nonl(void);
extern void noqiflush(void);
extern int noraw(void);
extern int notimeout(WINDOW *, bool);
extern int overlay(const WINDOW *, WINDOW *);
extern int overwrite(const WINDOW *, WINDOW *);
extern int pair_content(short, short *, short *);
extern int pechochar(WINDOW *, chtype);
extern int pnoutrefresh(WINDOW *, int, int, int, int, int);
extern int prefresh(WINDOW *, int, int, int, int, int);
extern int printw(char *, ...);
extern int putwin(WINDOW *, FILE *);
extern void qiflush(void);
extern int raw(void);
extern int redrawwin(WINDOW *);
extern int refresh(void);
extern int resetty(void);
extern int reset_prog_mode(void);
extern int reset_shell_mode(void);
extern int ripoffline(int, int (*_bkgrnd) (WINDOW *, int)) ;
extern int savetty(void);
extern int scanw(const char *, ...);
extern int scr_dump(const char *);
extern int scr_init(const char *);
extern int scroll(WINDOW *);
extern int scrollok(WINDOW *, bool);
extern int scr_restore(const char *);
extern int scr_set(const char *);
extern int setscrreg(int, int);
extern int SCREEN *set_term(SCREEN *);
extern int slk_attroff(const chtype);
extern int slk_attron(const chtype);
extern int slk_attrset(const chtype);
extern int slk_attr_set(const attr_t, short, void *);
extern int slk_clear(void);
extern int slk_color(short);
extern int slk_init(int);
extern char *slk_label(int);
extern int slk_noutrefresh(void);
extern int slk_refresh(void);
extern int slk_restore(void);
extern int slk_set(int, const char *, int);
extern int slk_touch(void);
extern int standout(void);
extern int standend(void);
extern int start_color(void);
extern WINDOW *subpad(WINDOW *, int, int, int, int);
extern WINDOW *subwin(WINDOW *, int, int, int, int);
extern int syncok(WINDOW *, bool);
extern chtype termattrs(void);
extern char *termname(void);
extern void timeout(int);
extern int typeahead(int);
extern int ungetch(int);
extern int untouchwin(WINDOW *);
extern void use_env(bool);
extern int vidattr(chtype);
extern int vidputs(chtype, int (*_bkgrnd) (int) );
extern int vline(chtype, int);
extern int vwprintw(WINDOW *, char *, va_list);
extern int vw_printw(WINDOW *, const char *, va_list);
extern int vw_scancw(WINDOW *, const char *, va_list);
extern int waddch(WINDOW *, const chtype);
extern int waddchnstr(WINDOW *, const chtype *, int);
extern int waddstr(WINDOW *, const char *);
extern int wattron(WINDOW *, int);
extern int wattroff(WINDOW *, int);
extern int wattr_set(WINDOW *, attr_t, short, void *);
extern int wbkgd(WINDOW *, chtype);
extern void wbkgdset(WINDOW *, chtype);
extern int wborder(WINDOW *, chtype, chtype, chtype, chtype, chtype, chtype, chtype, chtype);
extern int wchgat(WINDOW *, int, attr_t, short, const void *);
extern int wclear(WINDOW *);
extern int wclrbot(WINDOW *);
extern int wclearoef(WINDOW *);
extern int wcolor_set(WINDOW *, int, attr_t, short, const void *);
extern int wcursyncup(WINDOW *);
extern int wdelch(WINDOW *);
extern int wdeletein(WINDOW *);
extern int wechochar(WINDOW *, int);
extern int werase(WINDOW *);
extern int wgetch(WINDOW *);
extern int wgetnstr(WINDOW *, char *, int);
extern int wgetstr(WINDOW *, int);
extern int whline(WINDOW *, chtype, int);
extern int wchar(WINDOW *, int);
extern int winchnstr(WINDOW *, chtype, int);
extern int winchstr(WINDOW *, int);
extern int winsnstr(WINDOW *, const char *, int);
extern int winsnstr(WINDOW *, const char *);
extern int wstandend(WINDOW *);
extern void wsyncdown(WINDOW *);
extern void wsyncup(WINDOW *);
extern int wtimeout(WINDOW *, int);
extern int wtouchln(WINDOW *, int, int, int);
extern int wvline(WINDOW *, chtype, int);
extern char *unctrl(chtype);
extern int COLORS;
extern int COLOR_PAIRS;
extern chtype acs_map[];
extern WINDOW *curscr;
extern WINDOW *stdscr;
extern int COLS;
extern int LINES;
extern int touchline(WINDOW *, int, int);
extern int touchwin(WINDOW *);

14.6.2 term.h

extern int putp(const char *);
extern int tigetflag(const char *);
extern int tigetnum(const char *);
extern char *tigetstr(const char *);
extern char *tparm(const char *, ...);
extern TERMINAL *set_curterm(TERMINAL *);
extern int del_curterm(TERMINAL *);
extern int restartterm(char *, int, int *);
extern int setupterm(char *, int, int *);
extern char *tgetstr(char *, char **);
extern char *tgoto(const char *, int, int);
extern int tgetent(char *, const char *);
extern char *tgetflag(char *);
extern int tgetnum(char *);
extern int tputs(const char *, int, int (*)(int) );
extern TERMINAL *cur_term;

14.7 Interfaces for libutil

Table 14-6 defines the library name and shared object name for the libutil library

<table>
<thead>
<tr>
<th>Library:</th>
<th>libutil</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libutil.so.1</td>
</tr>
</tbody>
</table>

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

[LSB] This Specification

14.7.1 Utility Functions

14.7.1.1 Interfaces for Utility Functions

An LSB conforming implementation shall provide the generic functions for Utility Functions specified in Table 14-7, with the full mandatory functionality as described in the referenced underlying specification.
Table 14-7 libutil - Utility Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>forkpty</td>
<td>LSB</td>
</tr>
<tr>
<td>login</td>
<td>LSB</td>
</tr>
<tr>
<td>login_tty</td>
<td>LSB</td>
</tr>
<tr>
<td>logout</td>
<td>LSB</td>
</tr>
<tr>
<td>logwtmp</td>
<td>LSB</td>
</tr>
<tr>
<td>openpty</td>
<td>LSB</td>
</tr>
</tbody>
</table>

### 14.8 Interface Definitions for libutil

The interfaces defined on the following pages 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 in Section 14.7 shall behave as described in the referenced base document.

#### forkpty

**Name**

forkpty — Create a new process attached to an available pseudo-terminal

**Synopsis**

```c
#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 examining 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 initialized 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 initialized 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 initialized 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 field 
shall 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.

Note: 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 *slave, 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 slave 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.
V Commands and Utilities
15 Commands and Utilities

15.1 Commands and Utilities

An LSB conforming implementation shall provide the commands and utilities as described in Table 15-1, with at least the behavior described as mandatory in the referenced 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, by default, GNU option parsing differs from POSIX, unless the environment variable POSIXLY_CORRECT is set. For example, ls . -a in GNU ls means to list the current directory, showing all files (that is, "." is an operand and -a is an option). In POSIX, "." and -a are both operands, and the command means to list the current directory, and also the file named -a. Suggesting that applications rely on the setting of the POSIXLY_CORRECT environment variable, or try to set it, seems worse than just asking the applications to invoke commands in ways which work with either the POSIX or GNU behaviors.

Table 15-1 Commands And Utilities

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
### 15 Commands and Utilities

<table>
<thead>
<tr>
<th>Command</th>
<th>[1]</th>
<th>[2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>crontab</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>groupdel</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>make</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>renice</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>uniq</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>csplit</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>groupmod</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>man</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>rm</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>useradd</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>cut</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>groups</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>md5sum</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>rmdir</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>userdel</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>gunzip</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>mkdir</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>sed</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>usermod</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>dd</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>gzip</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>mkfifo</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>sendmail</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>wc</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>diff</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>hostname</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>mktemp</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>shutdown</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>zcat</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>dirname</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>iconv</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>mknod</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>sh</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>xargs</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>df</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>head</td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>mktemp</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>sh</td>
<td>[2]</td>
<td></td>
</tr>
<tr>
<td>xargs</td>
<td>[2]</td>
<td></td>
</tr>
</tbody>
</table>

### Referenced Specification(s)


[2]. This Specification

An LSB conforming implementation shall provide the shell built in utilities as described in Table 15-2, with at least the behavior described as mandatory in the referenced underlying specification, with the following exceptions:

1. The built in commands and utilities shall be provided by the `sh` utility itself, and need not be implemented in a manner so that they can be accessed via the exec family of functions as defined in ISO POSIX (2003) and should not be invoked directly by those standard utilities that execute other utilities (`env`, `find`, `nice`, `nohup`, `time`, `xargs`).

**Rationale (Informative):** Since the built in utilities must affect the environment of the calling process, they have no effect when executed as a file.

### Table 15-2 Built In Utilities

<table>
<thead>
<tr>
<th>Command</th>
<th>[1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd</td>
<td>[1]</td>
</tr>
<tr>
<td>getopts</td>
<td>[1]</td>
</tr>
<tr>
<td>read</td>
<td>[1]</td>
</tr>
<tr>
<td>umask</td>
<td>[1]</td>
</tr>
<tr>
<td>wait</td>
<td>[1]</td>
</tr>
</tbody>
</table>

### Referenced Specification(s)


### 15.2 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 (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

Options

-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.

Optional Control Files
The implementation shall support the XSI optional behavior for access control; however the files at.allow and at.deny may 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 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 differences as listed below.

Optional Control Files
The implementation shall support the XSI optional behavior for access control; however the files at.allow and at.deny may 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 extensions as listed below.

Extensions
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.
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 SUSv2 but with differences as listed below.

Differences
The -p option has unspecified behavior.

Note: Although col is shown as legacy in SUSv2, it is not (yet) deprecated in the
LSB.
cpio

**Name**
cpio — copy file archives in and out

**Description**
cpio is as specified in SUSv2, but with differences as listed below.

**Differences**
Some elements of the Pattern Matching Notation are optional; see 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.

**Optional Control Files**
The implementation shall support the XSI optional behavior for access control; however the files cron.allow and cron.deny may 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 file system disk space usage

Description

The df command shall behave as specified in ISO POSIX (2003), but with differences as listed below.

Differences

Options

If the \(-k\) option is not specified, disk space is shown in unspecified units. If the \(-P\) option is specified, the size of the unit shall be printed on the header line in the format "%4s-blocks". Applications should specify \(-k\).

The XSI option \(-t\) has unspecified behavior. Applications should not specify \(-t\).

Rationale: The most common implementation of df uses the \(-t\) option for a different purpose (restricting output to a particular file system type), and use of \(-t\) is therefore non-portable.

Operand May Identify Special File

If an argument is the absolute file name of a special file containing a mounted file system, df shall show the space available on that file system rather than on the file system containing the special file (which is typically the root file system).

Note: In ISO POSIX (2003) the XSI optional behavior permits an operand to name a special file, but appears to require the operation be performed on the file system containing the special file. A defect report has been submitted for this case.
**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.

**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 — write arguments to standard output

Synopsis
echo [string...]

Description
The echo command is as specified in ISO POSIX (2003), but with the following differences.

Implementations may support implementation-defined options to echo. The behavior of echo if any arguments contain backslashes is also implementation defined.

Application Usage
Conforming applications should not run echo with a first argument starting with a hyphen, or with any arguments containing backslashes; they should use printf in those cases.

Note: The behavior specified here is similar to that specified by ISO POSIX (2003) without the XSI option. However, the LSB strongly recommends conforming applications not use any options (even if the implementation provides them) while ISO POSIX (2003) specifies behavior if the first operand is the string -n.

egrep

Name
egrep — search a file with an Extended Regular Expression 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 

-find

Pattern Matching
Some elements of the Pattern Matching Notation are optional; see Pattern Matching Notation.

Option and Operand Handling
Options and operands to find shall behave as described in ISO POSIX (2003), except as follows:


-exec ... +

argument aggregation need not be supported

Rationale: The 

-find

options are not yet widely available in implementations of the find command, nor is argument aggregation. A future version of this specification will require these features be supported.
**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**

The *fuser* command is a system administration utility, see Path For System Administration Utilities.

**Option 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
Certain aspects of regular expression matching are optional; see Regular Expressions.
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.

The groupadd command is a system administration utility, see Path For System Administration Utilities.

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.

The groupdel command is a system administration utility, see Path For System Administration Utilities.
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.

The groupmod command is a system administration utility, see Path For System Administration Utilities.

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.

  Note: 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. Filesystem Hierarchy Standard requires that if gunzip exists, it must be a symbolic or hard link to /bin/gzip. This specification additionally allows gunzip to be a wrapper script which calls gzip -d.
gzip

Name
gzip — compress or expand files

Synopsis
gzip [-cdfhLlnRtvV] [-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.


Options
-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 output: 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, ratio and uncompressed name for each compressed file. For files that are not in gzip format, the uncompressed size shall be given as -l. If the --verbose or -v option is also specified, the crc and timestamp for the uncompressed file shall also be displayed.

For decompression, gzip shall support at least the following compression methods:
• deflate (RFC 1951: DEFLATE Compressed Data Format Specification)
• compress (ISO POSIX (2003))
The crc shall be given as ffffffff for a file not in gzip format.
If the \texttt{--name} or \texttt{-N} option is also specified, the uncompressed name, date and time are those stored within the compressed file, if present.

If the \texttt{--quiet} or \texttt{-q} option is also specified, the title and totals lines are not displayed.

\texttt{-L, --license}

displays the \texttt{gzip} license and quit.

\texttt{-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.

\texttt{-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.

\texttt{-q, --quiet}

suppresses all warnings.

\texttt{-r, --recursive}

travels the directory structure recursively. If any of the file names specified on the command line are directories, \texttt{gzip} will descend into the directory and compress all the files it finds there (or decompress them in the case of \texttt{gunzip}).

\texttt{-S .suf, --suffix .suf}

uses suffix \texttt{.suf} instead of \texttt{.gz}.

\texttt{-t, --test}

checks the compressed file integrity.

\texttt{-v, --verbose}

displays the name and percentage reduction for each file compressed or decompressed.

\texttt{-#, --fast, --best}

regulates the speed of compression using the specified digit \#, where \texttt{-1} or \texttt{--fast} indicates the fastest compression method (less compression) and \texttt{-9} or \texttt{--best} indicates the slowest compression method (best compression). The default compression level is \texttt{-6} (that is, biased towards high compression at expense of speed).

\section*{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.

Note: 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 DEST 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
        behave as numbered if numbered backups exist, or simple otherwise.
    simple or never
        append a suffix to the name. The default suffix is ' ~', but can be overridden 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 — activate an init script

Synopsis
/usr/lib/lsb/install_initd initd_file

Description
install_initd shall activate a system initialization file that has been copied to an implementation defined location such that this file shall be run at the appropriate point during system initialization. The install_initd command is typically called in the postinstall script of a package, after the script has been copied to /etc/init.d. See also Installation and Removal of Init Scripts.
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.

Note: Although this command has many similarities with the optional ipcs utility described in ISO POSIX (2003), it has substantial differences and is therefore described separately. The options specified here have similar meaning to those in ISO POSIX (2003); other options specified there have unspecified behavior on an LSB conforming implementation. See Application Usage below. The output format is not specified.

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 implementations of ipcs provide 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 formatting flags may be used. At least one of the -t, -c and -p options and at least one of -m, -q and -s options shall be specified. If no options are specified, the output is unspecified.
**killall**

**Name**

`killall` — kill processes by name

**Synopsis**

```
killall [-egiqvw] [-signal] name...
```

**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.

- `-v`
  
  reports if the signal was successfully sent.

**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
  displays version information.

**lpr**

**Name**

lpr — offline print

**Synopsis**

```bash
```

**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 shall behave as specified in ISO POSIX (2003), but with extensions listed below.

Extensions

-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.

Note: The LSB does not specify the meaning of the major and minor devices numbers.

-p
in addition to ISO POSIX (2003) XSI optional behavior of printing a slash for a directory, ls -p may display other characters for other file types.
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 separated list of LSB module descriptions. LSB module descriptions are dash separated tuples containing the module name, version, and architecture name. The output is a single line of text of the following format:

  ```
  LSB Version:	ListAsDescribedAbove
  ```

  **Note:** An implementation may support multiple releases of the same module. Version specific library interfaces, if any, will be selected by the program interpreter, which changes from release to release. Version specific commands and utilities, if any, will be described in the relevant specification.

- `-i`, `--id`
  
  displays string id of distributor. The output is a single line of text of the following format:

  ```
  Distributor ID:	DistributorID
  ```

- `-d`, `--description`
  
  displays single line text description of distribution. The output is of the following format:

  ```
  Description:	Description
  ```

- `-r`, `--release`
  
  displays release number of distribution. The output is a single line of text of the following format:

  ```
  Release:	Release
  ```

- `-c`, `--codename`
  
  displays codename according to distribution release. The output is a single line of text of the following format:

  ```
  Codename:	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.

Example
The following command will list the LSB Profiles which are currently supported
on this platform.

    example% lsb_release -v

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.

Note: This option may be deprecated in a future release of this specification.

If type is p, major and minor shall not be specified. Otherwise, these
parameters are mandatory.

Future Directions
This command may be deprecated in a future version of this specification. The
major and minor operands are insufficiently portable to be specified usefully
here. Only a FIFO can be portably created by this command, and the mknod
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:
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]
```

**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 file systems (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 file system type. Not all file systems 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 unprivileged 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.

Note: 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.

--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 `msgid` 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 `msgid`s until the next `domain` directive are put into the message object file domainname (or domainname.mo if `--strict` option is specified).

3. Duplicate `msgid`s 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 `msgid`s 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 \( n = 0, 1, 2, \ldots \)) 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 15-1 Escape Sequences**
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.

**Note:** 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 `msgid` 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**
  
  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 the 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=code set 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 string's codeset to the output string's codeset will be performed upon the call of gettext(), dgettext(), dcgettext(), ngettext(), dngettext() or 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"
The following command will produce the output files messages.mo, help_domain.mo, and error_domain.mo.

example% msgfmt module1.po module2.po
example% msgfmt module1.po module2.po module3.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 –i 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**

The *od* command shall provide all of the mandatory functionality specified in ISO POSIX (2003), but with extensions and differences to the XSI optional behavior as listed below.

**Extensions and Differences**

- **-s**
  
  unspecified behavior.

  **Note:** Applications wishing to achieve the ISO POSIX (2003) behavior for `-s` should instead use `-t d2`.

- **-w width, --width=width**

  each output line is limited to *width* bytes from the input.

- **--traditional**

  accepts arguments in traditional form, see Traditional Usage below.

  **Note:** The XSI optional behavior for offset handling described in ISO POSIX (2003) is not supported unless the `--traditional` option is also specified.

**Pre-POSIX and XSI Specifications**

The LSB supports mixing options between the mandatory and XSI optional synopsis forms in ISO POSIX (2003). The LSB shall support the following 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

is equivalent to -t d2, selects decimal two byte units.

Note: This usage may change in future releases; portable applications should use -t d2.

-l

is equivalent to -t d4, selects decimal longs.

-o

is equivalent to -t o2, selects octal two byte units.

-x

is equivalent to -t x2, selects hexadecimal two byte units.

Note that the XSI option -s need not be supported.

Traditional Usage

If the --traditional option is specified, there may be between zero and three operands specified.

If no operands are specified, then od shall read the standard input.

If there is exactly one operand, and it is an offset of the form [+]-offset[.]b, then it shall be interpreted as specified in ISO POSIX (2003). The file to be dumped shall be the standard input.

If there are exactly two operands, and they are both of the form [+]-offset[.]b, then the first shall be treated as 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 output line produced for each input block shall be preceded by the input offset, cumulative across input files, of the next byte to be written, followed by the label, in parentheses. The label shall increment in the same manner as the offset.

If there are three operands, then the first shall be the file to dump, the second the offset, and the third the label.

Note: Recent versions of coreutils contain an od utility that conforms to ISO POSIX (2003). However, in April 2005, this version was not in widespread use. A future version of this specification may remove the differences.
**passwd**

**Name**

*passwd* – change user password

**Synopsis**

```
passwd [-x max] [-n min] [-w warn] [-i inact] name passwd [-l | -u] name
```

**Description**

*passwd* changes authentication information for user and group accounts, including passwords and password expiry details, and may be used to enable and disable accounts. Only a user with appropriate privilege may change the password for other users or modify the expiry information.

**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.

The pidof command is a system administration utility, see Path For System Administration Utilities.

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 init 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 init 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 script from the repository. See also Installation and Removal of Init Scripts.

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 Regular Expressions.
sendmail

Name
sendmail — an electronic mail transport agent

Synopsis
/usr/sbin/sendmail [options] [address...]

Description
To deliver electronic mail (email), applications shall support the interface provided by 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.

If an address is preceded by a backslash, '\', it is unspecified if the address is subject to local alias expansion.

The format of messages shall be as defined in RFC 2822:Internet Message Format.

Note: The name sendmail was chosen for historical reasons, but the sendmail command specified here is intended to reflect functionality provided by smail, exim and other implementations, not just the sendmail implementation.

Options
-bm
read mail from standard input and deliver it to the recipient addresses. This is the default mode of operation.

-bp
If the user has sufficient privilege, list information about messages currently in the mail queue.

-bs
use the SMTP protocol as described in RFC 2821:Simple Mail Transfer Protocol; read SMTP commands on standard input and write SMTP responses on standard output.

In this mode, sendmail shall accept \r\n (CR-LF), as required by RFC 2821:Simple Mail Transfer Protocol, and \n (LF) line terminators.

-F fullname
explicitly set 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 configuration of the agent.
-f name

explicitly set 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 shall be indicated in the message.

-i

ignore dots alone on lines by themselves in incoming messages. If this
options is not specified, a line consisting of a single dot shall terminate the
input. If -bs is also used, the behavior is unspecified.

-odb

deliver any mail in background, if supported; otherwise ignored.

-odf

deliver any mail in foreground, if supported; otherwise ignored.

-oem or -em

mail errors back to the sender. (default)

-oep or -ep

write errors to the standard error output.

-oeq or -eq

do not send notification of errors to the sender. This only works for mail
delivered locally.

-oi

is equivalent to -i.

-om

indicate 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

read 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: Internet Message Format.
If there are any operands, the recipients list is unspecified.
This option may be ignored when not in -bm mode (the default).

**Note:** It is recommended that applications use as few options as necessary, none if possible.

**Exit status**

0
successful completion on all addresses. This does not indicate successful delivery.

>0

there was an error.

**sh**

**Name**

*sh* — shell, the standard command language interpreter

**Description**

The *sh* utility shall behave as specified in ISO POSIX (2003), but with extensions listed below.

**Shell Invocation**

The shell shall support an additional option, `-l` (the letter *ell*). If the `-l` option is specified, or if the first character of argument zero (the command name) is a `-`, this invocation of the shell is a login shell.

An interactive shell, as specified in ISO POSIX (2003), that is also a login shell, or any shell if invoked with the `-l` option, shall, prior to reading from the input file, first read and execute commands from the file `/etc/profile`, if that file exists, and then from a file called `~/.profile`, if such a file exists.

**Note:** This specification requires that the *sh* utility shall also read and execute commands in its current execution environment from all the shell scripts in the directory `/etc/profile.d`. Such scripts are read and executed as a part of reading and executing `/etc/profile`. 
shutdown

Name
shutdown — shut the system down

Synopsis
/sbin/shutdown [-t sec] [-h | -r] [-akfF] time [warning-
message]/sbin/shutdown -c [warning-message]

Description
The shutdown command shall shut the system down in a secure way (first
synopsis), or cancel a pending shutdown (second synopsis). When the
shutdown is initiated, all logged-in users shall be notified immediately that the
system is going down, and users shall be prevented from logging in to the
system. The time specifies when the actual shutdown shall commence. See
below for details. At the specified time all processes are first notified that the
system is going down by the signal SIGTERM. After an interval (see -t) all
processes shall be sent the signal SIGKILL. If neither the -h or the -r argument
is specified, then the default behavior shall be to take the system to a runlevel
where administrative tasks can be run. See also Run Levels.

Note: This is sometimes referred to as "single user mode".

The -h and -r options are mutually exclusive. If either the -h or -r options are
specified, the system shall be halted or rebooted respectively.

Standard Options

-a
use access control. See below.

-t sec
tell the system to wait sec seconds between sending processes the warning
and the kill signal, before changing to another runlevel. The default period
is unspecified.

-k
do not really shutdown; only send the warning messages to everybody.

-r
reboot after shutdown.

-h
halt after shutdown. Actions after halting are unspecified (e.g. power off).

-f
advise the system to skip file system consistency checks on reboot.

-F
advise the system to force file system consistency checks on reboot.

-c
cancel an already running `shutdown`.

time

specify when to shut down.

The time argument shall have the following format: `[now | [+ ]mins | hh:mm]` If the format is `hh:mm`, `hh` shall specify the hour (1 or 2 digits) and `mm` is the minute of the hour (exactly two digits), and the shutdown shall commence at the next occurrence of the specified time. If the format is `mins` (or `+mins`), where `mins` is a decimal number, shutdown shall commence in the specified number of minutes. The word `now` is an alias for `+0`.

warning-message

specify a message to send to all users.

Access Control

If the `shutdown` utility is invoked with the `-a` option, it shall check that an authorized user is currently logged in on the system console. Authorized users are listed, one per line, in the file `/etc/shutdown.allow`. Lines in this file that begin with a `#' or are blank shall be ignored.

**Note:** The intent of this scheme is to allow a keyboard sequence entered on the system console (e.g. CTRL-ALT-DEL, or STOP-A) to automatically invoke `shutdown -a`, and can be used to prevent unauthorized users from shutting the system down in this fashion.
su

Name

su – change user ID

Synopsis

su [options] [-] [username [ARGS]]

Description

The su command shall start a shell running with the real and effective user and group IDs of the user username. If username is not specified, su shall default to an unspecified user with all appropriate privileges. If the -s or --shell is not specified, the shell to be invoked shall be that specified for username in the user database (see getpwnam(2)), or /bin/sh if there is no shell specified in the user database.

If the - option is specified, or if the first operand is -, the environment for the shell shall be initialized as if the new shell was a login shell (see Shell Invocation).

If the invoking user does not have appropriate privileges, the su command shall prompt for a password and validate this before continuing. Invalid passwords shall produce an error message. The su command shall log in an unspecified manner all invokations, whether successful or unsuccessful.

Any operands specified after the username shall be passed to the invoked shell.

If the option - is not specified, and if the first operand is not -, the environment for the new shell shall be initialized from the current environment. If none of the -m, -p, or --preserve-environment options are specified, the environment may be modified in unspecified ways before invoking the shell. If any of the -m, -p, or --preserve-environment options are specified, the environment shall not be altered.

Note: Although the su command shall not alter the environment, the invoked shell may still alter it before it is ready to interpret any commands.

Standard Options

- 

the invoked shell shall be a login shell.

-c command, --command=command

Invoke the shell with the option -c command.

-m, -p, --preserve-environment

The current environment shall be passed to the invoked shell. If the environment variable SHELL is set, it shall specify the shell to invoke, if it matches an entry in /etc/shells. If there is no matching entry in /etc/shells, this option shall be ignored if the - option is also specified, or if the first operand is -.

-s shell, --shell=shell

Invoke shell as the command interpreter. The shell specified shall be present in /etc/shells.
sync

Name
sync — flush file system 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
Some elements of the Pattern Matching Notation are optional; see 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 file system.

- 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, and with appropriate privilege, 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, with appropriate privilege, update the default values from the command line. If no options are specified, useradd displays the current default values.

The useradd command is a system administration utility, see Path For System Administration Utilities.

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()`. The default is to disable the account.

\(-r\)

creates a system account, that is, a user with a User ID in the range reserved for system account users. If there is not a User ID 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.

The userdel command is a system administration utility, see Path For System Administration Utilities.

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

Description

The usermod command shall modify an entry in the user account database.

The usermod command is a system administration utility, see Path For System Administration Utilities.

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.

  **Note:** These options have been implemented in findutils-4.2.9, but this version of the utilities is not in widespread use as of April 2005. However, future versions of this specification will require support for these arguments.

**zcat**

**Name**

zcat — uncompress files to standard output

**Description**

The zcat utility shall behave as described in ISO POSIX (2003), with differences listed below.

The Filesystem Hierarchy Standard requires that if zcat exists, it must be a symbolic or hard link to /bin/gzip. This specification additionally allows zcat to be a wrapper script which calls gzip -c -d.

**Differences**

The zcat utility shall write to standard output the uncompressed form of files that have been compressed using any of the compression methods supported by the gzip utility. It is the equivalent of gzip -c -d. Input files are not affected.
VI Execution Environment
16 File System Hierarchy

An LSB conforming implementation shall provide the mandatory portions of the file system 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 file system.

The FHS specifies certain behaviors for a variety of commands if they are present (for example, ping or python). However, LSB conforming 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

16.1 /dev: Device Files

The devices described in Chapter 6. "Operating System Specific Annex", Section 6.1. "Linux", subsection 6.1.3. "/dev: Devices and special files" in the Filesystem Hierarchy Standard are required on an LSB conforming system. 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.

16.2 /etc: Host-specific system configuration

In addition to the requirements for /etc in the Filesystem Hierarchy Standard, an LSB conforming system shall also provide the following directories or symbolic links to directories:

/etc/cron.d

A directory containing extended crontab files; see Cron Jobs.

/etc/cron.daily

A directory containing shell scripts to be executed once a day; see Cron Jobs.

/etc/cron.hourly

A directory containing shell scripts to be executed once per hour; see Cron Jobs.

/etc/cron.monthly

A directory containing shell scripts to be executed once per month; see Cron Jobs.
16 File System Hierarchy

/\etc/cron.weekly
A directory containing shell scripts to be executed once a week; see Cron Jobs.

/\etc/init.d
A directory containing system initialization scripts; see Installation and Removal of Init Scripts.

/\etc/profile.d
A directory containing shell scripts. Script names should follow the same conventions as specified for cron jobs (see Cron Jobs, but should have the suffix .sh. The behavior is unspecified if a script is installed in this directory that does not have the suffix .sh.

The sh utility shall read and execute commands in its current execution environment from all the shell scripts in this directory that have the suffix .sh when invoked as an interactive login shell, or if the -l (the letter ell) is specified (see Shell Invocation).

Future Directions: These directories are required at this version of the LSB since there is not yet an agreed method for abstracting the implementation so that applications need not be aware of these locations during installation. However, Future Directions describes a tool, lsbinstall, that will make these directories implementation specific and no longer required.

16.2.1 File Naming Conventions
Conforming implementations and applications installing files into any of the above locations under /\etc may only use filenames from the following managed namespaces:

• Assigned names. Such names must be chosen from the character set [a-z0-9]. In order to avoid conflicts these 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).

  Note: Commonly used names should be reserved in advance; developers for projects are encouraged to reserve names from LANANA, so that each distribution can use the same name, and to avoid conflicts with other projects.

• Hierarchical names. Script names in this category take the form: <hier1>-<hier2>-...<name>, where name is 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 names. Names that begin with the character '_' are reserved for distribution use only. These names should be used for essential system packages only.

  Note: A non-conforming application may still have polluted these managed namespaces with unregistered filenames; a conforming application should check for namespace collisions and take appropriate steps if they occur.

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 essential 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 system administrators may need to manually start and stop services. Given this, they should be standardized on a per-package basis. This is the rationale behind having the LANANA organization assign these names. The LANANA may be called upon to handle other namespace issues, such as package/prerequisites naming.

### 16.3 User Accounting Databases

The Filesystem Hierarchy Standard specifies two optional locations for user accounting databases used by the `getutent()`, `getutent_r()`, `getutxent()`, `getutxid()`, `getutxline()`, and `pututxline()` functions. These are `/var/run/utmp` and `/var/run/wtmp`.

The LSB does not specify the format or structure of these files, or even if they are files at all. They should be used only as "magic cookies" to the `utmpname()` function.

### 16.4 Path For System Administration Utilities

Certain utilities used for system administration (and other privileged commands) may be stored in `/sbin`, `/usr/sbin`, and `/usr/local/sbin`. Applications requiring to use commands identified as system administration utilities should add these directories to their PATH. By default, as described in ISO POSIX (2003), standard utilities shall be found on the PATH returned by `getconf PATH` (or `command -p getconf PATH` to be guaranteed to invoke the correct version of `getconf`).
17 Additional Recommendations

17.1 Recommendations for applications on ownership and permissions

17.1.1 Directory Write Permissions

The application should not depend on having directory write permission in any directory except /tmp, /var/tmp, and the invoking user's home directory.

In addition, the application may store variable data in /var/opt/package, (where package is the name of the application package), if such a directory is created with appropriate permissions during the package installation.

For these directories the application should be able to work with directory write permissions restricted by the S_ISVTXT bit, implementing the restricted deletion mode as described for the XSI option for ISO POSIX (2003).

17.1.2 File Write Permissions

The application should not depend on file write permission to any file that it does not itself create.

17.1.3 File Read and execute Permissions

The application should not depend on having read permission to every file and directory.

17.1.4 SUID and SGID Permissions

The application should not depend on the set user ID or set group ID (the S_ISUID or S_ISGID permission bits) 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.

17.1.5 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.
17.1.6 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.

17.1.7 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.

17.1.8 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.
18 Additional Behaviors

18.1 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.

Note: 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

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 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.

Note: Linux allows rename() on a directory without having write access, but the LSB does not require this.

18.1.1 Special Requirements

LSB conforming systems shall enforce certain special additional restrictions above and beyond those required by ISO POSIX (2003).

Note: 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 "a" shall be an invalid mode argument to popen().

The fcntl() function shall fail and set errno to EDEADLK if the cmd argument is F_SETLKW, and the lock is blocked by a lock from another process already blocked by the current process.

The opendir() function shall consume a file descriptor; the readdir() function shall fail and set errno to EBADF if the underlying file descriptor is closed.

The link() function shall not work across file systems, and shall fail and set errno to EXDEV as described as optional behavior in ISO POSIX (2003).
19 Localization

19.1 Introduction

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 ISO POSIX (2003). Other locales may be supported. Implementations may define additional locale categories not defined by that standard.

**Note:** Implementations choosing additional locale categories should be aware of ISO/IEC TR14652 and are advised not to choose names that conflict with that specification. If implementations provide locale categories whose names are part of the FDCC set of ISO/IEC TR14652, they should behave as defined by that specification.

19.2 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:

- Handling of a multi-character collating element is optional.
- This affects at least the following utilities:
  - `awk` (see awk)
  - `grep` (see grep) (including `egrep`, see egrep)
  - `sed` (see sed)
- It also affects the behavior of interfaces in the base libraries, including at least
  - `regexec()` (see regexec)

19.3 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:

- 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.
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) and tar (tar).
VII System Initialization
20 System Initialization

20.1 Cron Jobs

In addition to the individual user crontab files specified by ISO POSIX (2003), which are located in /var/spool/cron as specified by the Filesystem Hierarchy Standard (FHS), the process that executes scheduled commands shall also process the following additional crontab files, which are in a different format (see below). /etc/crontab, /etc/cron.d/*.

The installation of a package shall not modify the crontab file /etc/crontab, and shall not directly modify the user crontab files in /var/spool/cron/crontabs. but may use the crontab command to modify the latter.

If a package wishes to install a job that has to be executed periodically, it shall place an executable cron script in one of the following directories:

/etc/cron.hourly
/etc/cron.daily
/etc/cron.weekly
/etc/cron.monthly

As these directory names suggest, the files within them are executed on a hourly, daily, weekly, or monthly basis, respectively, under the control of an entry in one of the system crontab files, at an unspecified time of day. See below for the rules concerning the names of cron scripts.

Note: It is recommended that cron scripts installed in any of these directories be script files rather than compiled binaries so that they may be modified by the local system administrator. Conforming applications may only install cron scripts which use an interpreter required by this specification or provided by this or another conforming application.

This specification does not define the concept of a package upgrade. Implementations may do different things when packages are upgraded, including not replacing a cron script if it marked as a configuration file, particularly if the cron script appears to have been modified since installation. In some circumstances, the cron script may not be removed when the package is uninstalled. Applications should design their installation procedure and cron scripts to be robust in the face of such behavior. In particular, cron scripts should not fail obscurely if run in unexpected circumstances. Testing for the existence of application binaries before executing them is suggested.

Future versions of this specification may remove the need to install file directly into these directories, and instead abstract the interface to the cron utility in such a way as to hide the implementation. Please see Future Directions.

If a certain task has to be executed at other than the predefined frequencies, the package shall install a file /etc/cron.d/cron-name. The file shall have the same format as that described for the crontab command in ISO POSIX (2003), except that there shall be an additional field, username, before the name of the command to execute. For completeness, the seven fields shall be:

1. Minute [0,59]
2. Hour [0,23]
3. Day of the month [1,31]
4. Month of the year [1,12]
5. Day of the week [0,6] (with 0=Sunday)
6. Username
7. command [args ...]

This file shall be processed by the system automatically, with the named command being run at the specified time, as the specified username.

Applications installing files in these directories shall use the LSB naming conventions (see File Naming Conventions).

20.2 Init Script Actions

Conforming applications which need to execute commands on changes to the system run level (including boot and shutdown), may install one or more init scripts. Init scripts provided by conforming applications shall accept a single argument which selects the action:

- 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 actions shall be supported by all init scripts; the reload and the try-restart actions are optional. Other init-script actions may be defined by the init script.

Init scripts shall ensure that they will behave sensibly if invoked with start when the service is already running, or with stop when not running, and that they do not kill similarly-named user processes. The best way to achieve this is to use the init-script functions provided by /lib/lsb/init-functions (see Init Script Functions)

If a service reloads its configuration automatically (as in the case of cron, for example), the reload action of the init script shall behave as if the configuration was reloaded successfully. The restart, try-restart, reload and force-reload actions may be atomic; that is if a service is known not to be operational after a restart or reload, the script may return an error without any further action.

Note: This specification does not define the concept of a package upgrade. Implementations may do different things when packages are upgraded, including not replacing an init script if it is marked as a configuration file, particularly if the file appears to have been modified since installation. In some circumstances, the init script may not be removed when the package is uninstalled. Applications should design their installation procedure and init scripts to be robust in the face of such behavior. In particular, init scripts should not fail obscurely if run in unexpected circumstances. Testing for the existence of application binaries before executing them is suggested.

If the status action is requested, 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

For all other init-script actions, the init script shall return an exit status of zero if the action was 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 exit with a non-zero status code:

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 (see Init Script Functions) **log_success_msg()**, **log_failure_msg()** and **log_warning_msg()**. 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.

**Note:** Since init scripts may be run manually by a system administrator with non-standard environment variable values for PATH, USER, LOGNAME, etc., init scripts should not depend on the values of these environment variables. They should set them to some known/default values if they are needed.

### 20.3 Comment Conventions for Init Scripts

Conforming applications may install one or more init scripts. These init scripts must be activated by invoking the **install_initd** command. Prior to package removal, the changes applied by **install_initd** must be undone by invoking **remove_initd**. See Installation and Removal of Init Scripts for more details.
install_initd and remove_initd determine actions to take by decoding a specially formatted block of lines in the script. This block shall be delimited by the lines

```plaintext
### BEGIN INIT INFO
### END INIT INFO
```

The delimiter lines may contain trailing whitespace, which shall be ignored. All lines inside the block shall begin with a hash character '#' in the first column, so the shell interprets them as comment lines which do not affect operation of the script. The lines shall be of the form:

```plaintext
# {keyword}: arg1 [arg2...] 
```

with exactly one space character between the '#' and the keyword, with a single exception. In lines following a line containing the Description keyword, and until the next keyword or block ending delimiter is seen, a line where the '#' is followed by more than one space or a tab character shall be treated as a continuation of the previous line.

The information extracted from the block is used by the installation tool or the init-script system to assure that init scripts are run in the correct order. It is unspecified whether the information is evaluated only when install_initd runs, when the init scripts are executed, or both. The information extracted includes run levels, defined in Run Levels, and boot facilities, defined in Facility Names.

The following keywords, with their arguments, are defined:

- **Provides**: boot_facility_1 [boot_facility_2...]
  - boot facilities provided by this init script. When an init script is run with a start argument, the boot facility or facilities specified by the Provides keyword shall be deemed present and hence init scripts which require those boot facilities should be started later. When an init script is run with a stop argument, the boot facilities specified by the Provides keyword are deemed no longer present.

- **Required-Start**: boot_facility_1 [boot_facility_2...]
  - facilities which must be available during startup of this service. The init-script system should insure init scripts which provide the Required-Start facilities are started before starting this script.

- **Required-Stop**: boot_facility_1 [boot_facility_2...]
  - facilities which must be available during the shutdown of this service. The init-script system should avoid stopping init scripts which provide the Required-Stop facilities until this script is stopped.

- **Should-Start**: boot_facility_1 [boot_facility_2...]
  - facilities which, if present, should be available during startup of this service. This allows for weak dependencies which do not cause the service to fail if a facility is not available. The service may provide reduced functionality in this situation. Conforming applications should not rely on the existence of this feature.

- **Should-Stop**: boot_facility_1 [boot_facility_2...]
  - facilities which should be available during shutdown of this service.
Default-Start: run_level_1 [run_level_2...]  
Default-Stop: run_level_1 [run_level_2...]  

which run levels should by default run the init script with a start (stop) argument to start (stop) the services controlled by the init script.

For example, if a service should run in runlevels 3, 4, and 5 only, specify "Default-Start: 3 4 5" and "Default-Stop: 0 1 2 6".

Short-Description: short_description  
provide a brief description of the actions of the init script. Limited to a single line of text.

Description: multiline_description  
provide a more complete description of the actions of the init script. May span multiple lines. In a multiline description, each continuation line shall begin with a '#' followed by tab character or a '#' followed by at least two space characters. The multiline description is terminated by the first line that does not match this criteria.

Additional keywords may be defined in future versions of this specification. Also, implementations may define local extensions by using the prefix X-implementor. For example, X-RedHat-foobardecl, or X-Debian-xyzzydecl.

Example:

```
### BEGIN INIT INFO  
# Provides: lsb-ourdb  
# Required-Start: $local_fs $network $remote_fs  
# Required-Stop: $local_fs $network $remote_fs  
# Default-Start: 2 3 4 5  
# Default-Stop: 0 1 6  
# Short-Description: start and stop OurDB  
# Description: OurDB is a very fast and reliable database engine used for illustrating init scripts  
### END INIT INFO
```

The comment conventions described in this section are only required for init scripts installed by conforming applications. Conforming runtime implementations are not required to use this scheme in their system provided init scripts.

Note: This specification does not require, but is designed to allow, the development of a system which runs init scripts in parallel. Hence, enforced-serialization of scripts is avoided unless it is explicitly necessary.

20.4 Installation and Removal of Init Scripts

Conforming applications may install one or more initialization scripts (or init scripts). An init script shall be installed in /etc/init.d (which may be a symbolic link to another location), by the package installer.

Note: The requirement to install scripts in /etc/init.d may be removed in future versions of this specification. See Host-specific system configuration and Future Directions for further details.

During the installer's post-install processing phase the program /usr/lib/lsb/install_initd must be called to activate the init script. Activation consists of arranging for the init script to be called in the correct order on system run-level changes (including system boot and shutdown), based on
dependencies supplied in the init script (see Comment Conventions for Init Scripts). The `install_initd` command should be thought of as a wrapper which hides the implementation details; how any given implementation arranges for the init script to be called at the appropriate time is not specified.

Example: if an init script specified "Default-Start: 3 4 5" and "Default-Stop: 0 1 2 6", `install_initd` might create "start" symbolic links with names starting with 'S' in `/etc/rc3.d`, `/etc/rc4.d` and `/etc/rc5.d` and "stop" symbolic links with names starting with 'K' in `/etc/rc0.d`, `/etc/rc1.d`, `/etc/rc2.d` and `/etc/rc6.d`. Such a scheme would be similar to the System V Init mechanism, but is by no means the only way this specification could be implemented.

The `install_initd` command takes a single argument, the full pathname of the installed init script. The init script must already be installed in `/etc/init.d`. The `install_initd` command will not copy it there, only activate it once it has been installed. For example:

```
/usr/lib/lsb/install_initd /etc/init.d/example.com-coffee
```

The `install_initd` command shall return an exit status of zero if the init-script activation was successful or if the init script was already activated. If the dependencies in the init script (see Comment Conventions for Init Scripts) cannot be met, an exit status of one shall be returned and the init script shall not be activated.

When a software package is removed, `/usr/lib/lsb/remove_initd` must be called to deactivate the init script. This must occur before the init script itself is removed, as the dependency information in the script may be required for successful completion. Thus the installer's pre-remove processing phase must call `remove_initd`, and pass the full pathname of the installed init script. The package installer is still responsible for removing the init script. For example:

```
/usr/lib/lsb/remove_initd /etc/init.d/example.com-coffee
```

The `remove_initd` program shall return an exit status of zero if the init script has been successfully deactivated or if the init script is not activated. If another init script which depends on a boot facility provided by this init script is activated, an exit status of one shall be returned and the init script shall remain activated. The installer must fail on such an exit code so it does not subsequently remove the init script.

Note: This specification does not describe a mechanism for the system administrator to manipulate the run levels at which an init script is started or stopped. There is no assurance that modifying the comment block for this purpose will have the desired effect.

### 20.5 Run Levels

The following run levels are specified for use by the `Default-Start` and `Default-Stop` actions defined in Comment Conventions for Init Scripts as hints to the `install_initd` command. Conforming implementations are not required to provide these exact run levels or give them the meanings described here, and may map any level described here to a different level which provides the equivalent functionality. Applications may not depend on specific run-level numbers.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>halt</td>
</tr>
<tr>
<td>1</td>
<td>single user mode</td>
</tr>
<tr>
<td>2</td>
<td>multiuser with no network services exported</td>
</tr>
</tbody>
</table>
20 System Initialization

<table>
<thead>
<tr>
<th>Run Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>normal/full multiuser</td>
</tr>
<tr>
<td>4</td>
<td>reserved for local use, default is normal/full multiuser</td>
</tr>
<tr>
<td>5</td>
<td>multiuser with a display manager or equivalent</td>
</tr>
<tr>
<td>6</td>
<td>reboot</td>
</tr>
</tbody>
</table>

Note: These run levels were chosen as reflecting the most frequent existing practice, and in the absence of other considerations, implementors are strongly encouraged to follow this convention to provide consistency for system administrators who need to work with multiple distributions.

20.6 Facility Names

Boot facilities are used to indicate dependencies in initialization scripts, as defined in Comment Conventions for Init Scripts. Facility names are assigned to scripts by the Provides: keyword. Facility names that begin with a dollar sign ($) are reserved system facility names.

Note: Facility names are only recognized in the context of the init script comment block and are not available in the body of the init script. In particular, the use of the leading '$' character does not imply system facility names are subject to shell variable expansion, since they appear inside comments.

Conforming applications shall not provide facilities that begin with a dollar sign. Implementations shall provide the following facility names:

$local_fs
all local file systems are mounted

$network
basic networking support is available. Example: a server program could listen on a socket.

$named
IP name-to-address translation, using the interfaces described in this specification, are available to the level the system normally provides them. Example: if a DNS query daemon normally provides this facility, then that daemon has been started.

$portmap
daemons providing SunRPC/ONCRPC portmapping service as defined in RFC 1833: Binding Protocols for ONC RPC Version 2 (if present) are running.

$remote_fs
all remote file systems are available. In some configurations, file systems such as /usr may be remote. Many applications that require $local_fs will probably also require $remote_fs.

$syslog
system logger is operational.
\textbf{20 System Initialization}

\textit{Stime}

the system time has been set, for example by using a network-based time program such as \texttt{ntp} or \texttt{rdate}, or via the hardware Real Time Clock.

Other (non-system) facilities may be defined by other conforming applications. These facilities shall be named using the same conventions defined for naming init scripts (see Script Names). Commonly, the facility provided by a conforming init script will have the same name as the name assigned to the init script.

\textbf{20.7 Script Names}

Since init scripts live in a single directory, they must share a single namespace. To avoid conflicts, applications installing files in this directories shall use the LSB naming conventions (see File Naming Conventions).

\textbf{20.8 Init Script Functions}

Each conforming init script shall execute the commands in the file \texttt{/lib/lsb/init-functions} in the current environment (see shell special built-in command \texttt{dot}). This file shall cause the following shell script commands to be defined in an unspecified manner.

\begin{quote}
\textbf{Note}: This can be done either by adding a directory to the PATH variable which defines these commands, or by defining shell aliases or functions.
\end{quote}

Although the commands made available via this mechanism need not be conforming applications in their own right, applications that use them should only depend on features described in this specification.

Conforming scripts shall not specify the "exit on error" option (i.e. \texttt{set -e}) when sourcing this file, or calling any of the commands thus made available.

The \texttt{start_daemon}, \texttt{killproc} and \texttt{pidofproc} functions shall use the following algorithm for determining the status and the process identifiers of the specified program.

1. If the \texttt{-p pidfile} option is specified, and the named \texttt{pidfile} exists, a single line at the start of the \texttt{pidfile} shall be read. If this line contains one or more numeric values, separated by spaces, these values shall be used. If the \texttt{-p pidfile} option is specified and the named \texttt{pidfile} does not exist, the functions shall assume that the daemon is not running.

2. Otherwise, \texttt{/var/run/basename.pid} shall be read in a similar fashion. If this contains one or more numeric values on the first line, these values shall be used. Optionally, implementations may use unspecified additional methods to locate the process identifiers required.

The method used to determine the status is implementation defined, but should allow for non-binary programs.

\begin{quote}
\textbf{Note}: Commonly used methods check either for the existence of the \texttt{/proc/pid} directory or use \texttt{/proc/pid/exe} and \texttt{/proc/pid/cmdline}. Relying only on \texttt{/proc/pid/exe} is discouraged since this specification does not specify the existence of, or semantics for, \texttt{/proc}. Additionally, using \texttt{/proc/pid/exe} may result in a not-running status for daemons that are written in a script language.
\end{quote}

Conforming implementations may use other mechanisms besides those based on pidfiles, unless the \texttt{-p pidfile} option has been used. Conforming applications should not rely on such mechanisms and should always use a
pidfile. When a program is stopped, it should delete its pidfile. Multiple process identifiers shall be separated by a single space in the pidfile and in the output of pidofproc.

**start_daemon** [-f] [-n nicelevel] [-p pidfile] pathname [args...]

runs the specified program as a daemon. The **start_daemon** function 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. **start_daemon** shall 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]

The **killproc** function shall stop 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 SIGHUP after an unspecified 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. The **killproc** function shall 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

The **pidofproc** function shall return one or more process identifiers for a particular daemon using the algorithm given above. Only process identifiers of running processes should be returned. Multiple process identifiers shall be separated by a single space.

**Note:** A process may exit between **pidofproc** discovering its identity and the caller of **pidofproc** being able to act on that identity. As a result, no test assertion can be made that the process identifiers returned by **pidofproc** shall be running processes.

The **pidofproc** function shall 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

The **log_success_msg** function shall cause the system to write a success message to an unspecified log file. The format of the message is unspecified. The **log_success_msg** function may also write a message to the standard output.

**Note:** The message should be relatively short; no more than 60 characters is highly desirable.

**log_failure_msg** message

The **log_failure_msg** function shall cause the system to write a failure message to an unspecified log file. The format of the message is unspecified. The **log_failure_msg** function may also write a message to the standard output.

**Note:** The message should be relatively short; no more than 60 characters is highly desirable.
The log_warning_msg function shall cause the system to write a warning message to an unspecified log file. The format of the message is unspecified. The log_warning_msg function may also write a message to the standard output.

**Note:** The message should be relatively short; no more than 60 characters is highly desirable.
VIII Users & Groups
21 Users & Groups

21.1 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.

21.2 User & Group Names

Table 21-1 describes required mnemonic user and group names. This specification makes no attempt to numerically assign user or group identity numbers, with the exception that both the User ID and Group ID for the user root shall be equal to 0.

Table 21-1 Required User & Group Names

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>root</td>
<td>Administrative user with all appropriate privileges</td>
</tr>
<tr>
<td>bin</td>
<td>bin</td>
<td>Legacy User ID/Group IDa</td>
</tr>
<tr>
<td>daemon</td>
<td>daemon</td>
<td>Legacy User ID/Group IDb</td>
</tr>
</tbody>
</table>

Notes:

a  The bin User ID/Group ID is included for compatibility with legacy applications. New applications should no longer use the bin User ID/Group ID.

b  The daemon User ID/Group ID was used as an unprivileged User ID/Group ID for daemons to execute under in order to limit their access to the system. Generally daemons should now run under individual User ID/Group IDs in order to further partition daemons from one another.

Table 21-2 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 21-2 Optional User & Group Names

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>adm</td>
<td>adm</td>
<td>Administrative special privileges</td>
</tr>
<tr>
<td>lp</td>
<td>lp</td>
<td>Printer special privileges</td>
</tr>
<tr>
<td>sync</td>
<td>sync</td>
<td>Login to sync the system</td>
</tr>
<tr>
<td>shutdown</td>
<td>shutdown</td>
<td>Login to shutdown the system</td>
</tr>
</tbody>
</table>
User Group Comments

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>halt</td>
<td>halt</td>
<td>Login to halt the system</td>
</tr>
<tr>
<td>mail</td>
<td>mail</td>
<td>Mail special privileges</td>
</tr>
<tr>
<td>news</td>
<td>news</td>
<td>News special privileges</td>
</tr>
<tr>
<td>uucp</td>
<td>uucp</td>
<td>UUCP special privileges</td>
</tr>
<tr>
<td>operator</td>
<td>root</td>
<td>Operator special privileges</td>
</tr>
<tr>
<td>man</td>
<td>man</td>
<td>Man special privileges</td>
</tr>
<tr>
<td>nobody</td>
<td>nobody</td>
<td>Used by NFS</td>
</tr>
</tbody>
</table>

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 file creation mask (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 Filesystem Hierarchy Standard and should be obtained by the getpwnam(), getpwnam_r(), getpwent(), getpwuid(), and getpwuid_r() functions.

21.3 User ID Ranges

The system User IDs from 0 to 99 should be statically allocated by the system, and shall not be created by applications.

The system User IDs from 100 to 499 should be reserved for dynamic allocation by system administrators and post install scripts using useradd.

21.4 Rationale

The purpose of specifying optional users and groups is to reduce the potential for name conflicts between applications and distributions.
IX Package Format and Installation
22 Software Installation

22.1 Introduction

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).

Note: Supplying an RPM format package is encouraged because it makes systems easier to manage. This specification does not require the implementation to use RPM as the package manager; it only specifies the format of the package file.

Applications are also encouraged to uninstall cleanly.

A package in RPM format may include a dependency on the LSB Core and other LSB specifications, as described in Section 22.6. Packages that are not in RPM format may test for the presence of a conforming implementation by means of the `lsb_release` utility.

Implementations shall provide a mechanism for installing applications in this packaging format with some restrictions listed below.

Note: The implementation 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.

22.2 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>
<thead>
<tr>
<th>Table 22-1 RPM File Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead</td>
</tr>
<tr>
<td>Signature</td>
</tr>
<tr>
<td>Header</td>
</tr>
<tr>
<td>Payload</td>
</tr>
</tbody>
</table>

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 installed.

22.2.1 Lead Section

```c
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 relevant architecture specific part of ISO/IEC 23360.

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.

22.2.2 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 22-2 Signature Format

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Record</td>
<td></td>
</tr>
<tr>
<td>Array of Index Records</td>
<td></td>
</tr>
<tr>
<td>Store of Index Values</td>
<td></td>
</tr>
</tbody>
</table>
22.2.2.1 Header Record

```c
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.

22.2.2.2 Index Record

```c
struct rpmhdrindex {
    int tag;
    int type;
    int offset;
    int count;
};
```

**tag**

Value identifying the purpose of the data associated with this Index Record. The 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.

22.2.2.2.1 Index Type Values

The possible values for the *type* field are defined in this table.
### Table 22-3 Index Type values

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Size (in bytes)</th>
<th>Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM_NULL_TYPE</td>
<td>0</td>
<td>Not Implemented.</td>
<td></td>
</tr>
<tr>
<td>RPM_CHAR_TYPE</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RPM_INT8_TYPE</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RPM_INT16_TYPE</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>RPM_INT32_TYPE</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RPM_INT64_TYPE</td>
<td>5</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>RPM_STRING_TYPE</td>
<td>6</td>
<td>variable, NUL terminated</td>
<td>1</td>
</tr>
<tr>
<td>RPM_BIN_TYPE</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RPM_STRING_ARRAY_TYPE</td>
<td>8</td>
<td>Variable, sequence of NUL terminated strings</td>
<td>1</td>
</tr>
<tr>
<td>RPM_I18NSTRING_TYPE</td>
<td>9</td>
<td>variable, sequence of NUL terminated strings</td>
<td>1</td>
</tr>
</tbody>
</table>

The string arrays specified for entries of type `RPM_STRING_ARRAY_TYPE` and `RPM_I18NSTRING_TYPE` are vectors of strings in a contiguous block of memory, each element separated from its neighbors by a NUL character.

Index records with type `RPM_I18NSTRING_TYPE` shall always have a `count` of 1. The array entries in an index of type `RPM_I18NSTRING_TYPE` correspond to the locale names contained in the `RPMTAG_HDR_I18N_TABLE` index.

#### 22.2.2.2 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 22-4 Header Private Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_HEADERSIGNATURES</td>
<td>62</td>
<td>BIN</td>
<td>16</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_HEADERIMMUTABLE</td>
<td>63</td>
<td>BIN</td>
<td>16</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_HDR_I18N_TABLE</td>
<td>100</td>
<td>STRING_ARRAY</td>
<td>Optional</td>
<td></td>
</tr>
</tbody>
</table>

© 2007 Linux Foundation
22.2.2.3 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.

22.2.3 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 22-5 Signature Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMSIGTAG_SIZE</td>
<td>1000</td>
<td>INT32</td>
<td>1</td>
<td>Required</td>
</tr>
</tbody>
</table>
RPMSIGTAG_SIZE

This tag specifies the combined size of the Header and Payload sections.

RPMSIGTAG_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 22-6 Signature Digest Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMSIGTAG_PAYLOADSIZE</td>
<td>1007</td>
<td>INT32</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMSIGTAG_SIZE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPMSIGTAG_PAYLOADSIZE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RPMSIGTAG_SHA1

This index contains the SHA1 checksum of the entire Header Section, including the Header Record, Index Records and Header store.

RPMSIGTAG_MD5

This tag specifies the 128-bit MD5 checksum of the combined Header and Archive sections.

These values exist to provide authentication of the package.

Table 22-7 Signature Signing Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMSIGTAG_SHA1</td>
<td>269</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMSIGTAG_MD5</td>
<td>1004</td>
<td>BIN</td>
<td>16</td>
<td>Required</td>
</tr>
<tr>
<td>RPMSIGTAG_SHA1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPMSIGTAG_MD5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RPMSIGTAG_DSA

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.

RPMSIGTAG_RSA

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.

RPMSIGTAG_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.

RPMSIGTAG_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.

22.2.4 Header Section

The Header section is implemented using the Header structure. The Header section defines the following additional tag values which may be used in the Index structures.

22.2.4.1 Package Information

The following tag values are used to indicate information that describes the package as a whole.

Table 22-8 Package Info Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_NAME</td>
<td>1000</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_VERS</td>
<td>1001</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_RELA</td>
<td>1002</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_SUMMARY</td>
<td>1004</td>
<td>I18NSTRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_DESCRIPI</td>
<td>1005</td>
<td>I18NSTRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_SIZE</td>
<td>1009</td>
<td>INT32</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_DISTRIB</td>
<td>1010</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_VEND</td>
<td>1011</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_LICENCE</td>
<td>1014</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_PACKAGER</td>
<td>1015</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_GROUP</td>
<td>1016</td>
<td>I18NSTRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_URL</td>
<td>1020</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_OS</td>
<td>1021</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
</tbody>
</table>
ISO/IEC 23360 Part 1:2007(E)  22 Software Installation

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_NAME</td>
<td>1022</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_VERSION</td>
<td>1044</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_ARCHIVESIZE</td>
<td>1046</td>
<td>INT32</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_RPMVERSION</td>
<td>1064</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_COOKIE</td>
<td>1094</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_DISTURL</td>
<td>1123</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_PAYLOADFORMAT</td>
<td>1124</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_PAYLOADCOMPRESSOR</td>
<td>1125</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_PAYLOADFLAGS</td>
<td>1126</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
</tbody>
</table>

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 description of the package.

RPMTAG_SIZE

This tag specifies the sum of the sizes of the regular files in the archive.

RPMTAG_DISTRIBUTION

A string containing the name of the distribution on which the package was built.

RPMTAG_VENDOR

A string containing the name of the organization that produced the package.
RPMTAG_LICENSE
This tag specifies the license which applies to this package.

RPMTAG_PACKAGE
A string identifying the tool used to build the package.

RPMTAG_GROUP
This tag specifies the administrative group to which this package belongs.

RPMTAG_URL
Generic package information URL.

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_DISTURL
URL for package.

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'.
## 22.2.4.2 Installation Information

The following tag values are used to provide information needed during the installation of the package.

### Table 22-9 Installation Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_PREIN</td>
<td>1023</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_POSTIN</td>
<td>1024</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_PREUN</td>
<td>1025</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_POSTUN</td>
<td>1026</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_PREINPROG</td>
<td>1085</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_POSTINPROG</td>
<td>1086</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_PREUNPROG</td>
<td>1087</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_POSTUNPROG</td>
<td>1088</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
</tbody>
</table>

- **RPMTAG_PREIN**
  
  This tag specifies the preinstall scriptlet. If present, then RPMTAG_PREINPROG shall also be present.

- **RPMTAG_POSTIN**
  
  This tag specifies the postinstall scriptlet. If present, then RPMTAG_POSTINPROG shall also be present.

- **RPMTAG_PREUN**
  
  This tag specifies the preuninstall scriptlet. If present, then RPMTAG_PREUNPROG shall also be present.

- **RPMTAG_POSTUN**
  
  This tag specifies the postuninstall scriptlet. If present, then RPMTAG_POSTUNPROG shall also be present.

- **RPMTAG_PREINPROG**
  
  This tag specifies the name of the interpreter to which the preinstall scriptlet will be passed. The interpreter pointed to by this index record shall be /bin/sh.

- **RPMTAG_POSTINPROG**
  
  This tag specifies the name of the interpreter to which the postinstall scriptlet will be passed. The interpreter pointed to by this index record shall be /bin/sh.
RPMTAG_PREUNPROG

This tag specifies the name of the interpreter to which the preuninstall scriptlet will be passed. The interpreter pointed to by this index record shall be /bin/sh.

RPMTAG_POSTUNPROG

This program specifies the name of the interpreter to which the postuninstall scriptlet will be passed. The interpreter pointed to by this index record shall be /bin/sh.

22.2.4.3 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 22-10 File Info Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_OLDFILENAMES</td>
<td>1027</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_FILESIZE</td>
<td>1028</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEMODES</td>
<td>1030</td>
<td>INT16</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILERDEVES</td>
<td>1033</td>
<td>INT16</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEMTIMES</td>
<td>1034</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEMD5S</td>
<td>1035</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILELINKTOS</td>
<td>1036</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEFLAGS</td>
<td>1037</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEUSERNAME</td>
<td>1039</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEGROUPNAME</td>
<td>1040</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEDEVICES</td>
<td>1095</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILEINODES</td>
<td>1096</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILELANGS</td>
<td>1097</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_DIRINDEXES</td>
<td>1116</td>
<td>INT32</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_BASE</td>
<td>1117</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Optional</td>
</tr>
</tbody>
</table>
RPMTAG_OLDFILENAMES

This tag specifies the filenames when not in a compressed format as determined by the absence 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_FILERDEVVS

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. See below.

RPMTAG_FILEUSERNAME

This tag specifies the owner of the corresponding file.

RPMTAG_FILEGROUPNAME

This tag specifies the group 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 system on the the system on which it was built.

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

One of RPMTAG_OLDFILENAMES or the tuple RPMTAG_DIRINDEXES, RPMTAG_BASENAMES, RPMTAG_DIRNAMES shall be present, but not both.

22.2.4.3.1 File Flags

The RPMTAG_FILEFLAGS tag value shall identify various characteristics of the file in the payload that it describes. It shall be an INT32 value consisting of either the value RPMFILE_NONE (0) or the bitwise inclusive or of one or more of the following values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMFILE_CONFIG</td>
<td>(1 &lt;&lt; 0)</td>
</tr>
<tr>
<td>RPMFILE_DOC</td>
<td>(1 &lt;&lt; 1)</td>
</tr>
<tr>
<td>RPMFILE_DONOTUSE</td>
<td>(1 &lt;&lt; 2)</td>
</tr>
<tr>
<td>RPMFILE_MISSINGOK</td>
<td>(1 &lt;&lt; 3)</td>
</tr>
<tr>
<td>RPMFILE_NOREPLACE</td>
<td>(1 &lt;&lt; 4)</td>
</tr>
<tr>
<td>RPMFILE_SPECFILE</td>
<td>(1 &lt;&lt; 5)</td>
</tr>
<tr>
<td>RPMFILE_GHOST</td>
<td>(1 &lt;&lt; 6)</td>
</tr>
<tr>
<td>RPMFILE_LICENSE</td>
<td>(1 &lt;&lt; 7)</td>
</tr>
<tr>
<td>RPMFILE_README</td>
<td>(1 &lt;&lt; 8)</td>
</tr>
<tr>
<td>RPMFILE_EXCLUDE</td>
<td>(1 &lt;&lt; 9)</td>
</tr>
</tbody>
</table>

These bits have the following meaning:

RPMTFILE_CONFIG

The file is a configuration file, and an existing file should be saved during a package upgrade operation and not removed during a package removal operation.

RPMTFILE_DOC

The file contains documentation.

RPMTFILE_DONOTUSE

This value is reserved for future use; conforming packages may not use this flag.
RPMFILE_MISSINGOK
The file need not exist on the installed system.

RPMFILE_NOREPLACE
Similar to the RPMFILE_CONFIG, this flag indicates that during an upgrade operation the original file on the system should not be altered.

RPMFILE_SPECFILE
The file is a package specification.

RPMFILE_GHOST
The file is not actually included in the payload, but should still be considered as a part of the package. For example, a log file generated by the application at run time.

RPMFILE_LICENSE
The file contains the license conditions.

RPMFILE_README
The file contains high level notes about the package.

RPMFILE_EXCLUDE
The corresponding file is not a part of the package, and should not be installed.

22.2.4.4 Dependency Information

The following tag values are used to provide information about interdependencies between packages.

Table 22-12 Package Dependency Tag Values

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_PROVIDENAME</td>
<td>1047</td>
<td>STRING_AR</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_REQUEST</td>
<td>1048</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_REQUIRENAME</td>
<td>1049</td>
<td>STRING_AR</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_REQUIREVERSION</td>
<td>1050</td>
<td>STRING_AR</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_CONFICTFLAGS</td>
<td>1053</td>
<td>INT32</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_CONFICTNAME</td>
<td>1054</td>
<td>STRING_AR</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_CONFICTVERSION</td>
<td>1055</td>
<td>STRING_AR</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_OBSOLETENAME</td>
<td>1090</td>
<td>STRING_AR</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_PROVIDE</td>
<td>1112</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
</tbody>
</table>
ISO/IEC 23360 Part 1:2007(E)

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDEFLAGS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RPMTAG_PROVIDEVERSION</td>
<td>1113</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_OBSOLETEFLAGS</td>
<td>1114</td>
<td>INT32</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_OBSOLETEVERSION</td>
<td>1115</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Optional</td>
</tr>
</tbody>
</table>

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 conflicting 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.
### 22.2.4.4.1 Package Dependency Values

The package dependencies are stored in the `RPMTAG_REQUIRENAME` and `RPMTAG_REQUIREVERSION` index records. The following values may be used.

#### Table 22-13 Index Type values

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Meaning</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpmlib(VersionedDependencies)</td>
<td>3.0.3-1</td>
<td>Indicates that the package contains <code>RPMTAG_PROVIDENAME</code>, <code>RPMTAG_OBSOLETENAME</code> or <code>RPMTAG_PREREQ</code> records that have a version associated with them.</td>
<td>Optional</td>
</tr>
<tr>
<td>rpmlib(PayloadFilesHavePrefix)</td>
<td>4.0-1</td>
<td>Indicates the filenames in the Archive have had &quot;.&quot; prepended to them.</td>
<td>Optional</td>
</tr>
<tr>
<td>rpmlib(CompressedFileNames)</td>
<td>3.0.4-1</td>
<td>Indicates that the filenames in the Payload are represented in the <code>RPMTAG_DIRINDEXES</code>, <code>RPMTAG_DIRNAME</code> and <code>RPMTAG_BASENAMES</code> indexes.</td>
<td>Optional</td>
</tr>
<tr>
<td>/bin/sh</td>
<td></td>
<td>Interpreter usually required for installation scripts.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Additional dependencies are specified in the Package Dependencies section of this specification, and in the relevant architecture specific part of ISO/IEC 23360.

### 22.2.4.4.2 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 22-14 Package Dependency Attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
### Name Value Meaning

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMSENSE_LESS</td>
<td>0x02</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_GREATER</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_EQUAL</td>
<td>0x08</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_PREREQ</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_INTERP</td>
<td>0x100</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_SCRIPT_PRE</td>
<td>0x200</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_SCRIPT_POS_T</td>
<td>0x400</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_SCRIPT_PRE_UN</td>
<td>0x800</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_SCRIPT_POS_TUN</td>
<td>0x1000</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_RPMLIB</td>
<td>0x1000000</td>
<td></td>
</tr>
</tbody>
</table>

### 22.2.4.5 Other Information

The following tag values are also found in the Header section.

**Table 22-15 Other Tag Values**

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag Value</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_BUILDDATE</td>
<td>1006</td>
<td>INT32</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_BUILDDHOST</td>
<td>1007</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_FILEVERIFYFLAGS</td>
<td>1045</td>
<td>INT32</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_CHANGELOGTIME</td>
<td>1080</td>
<td>INT32</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_CHANGELOGNAME</td>
<td>1081</td>
<td>STRING_AR</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_CHANGELOGTEXT</td>
<td>1082</td>
<td>STRING_AR</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_OPTFLAGS</td>
<td>1122</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_RHNPLATFORM</td>
<td>1131</td>
<td>STRING</td>
<td>1</td>
<td>Deprecated</td>
</tr>
<tr>
<td>RPMTAG_PLATFORM</td>
<td>1132</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
</tbody>
</table>

**RPMTAG_BUILDDATE**

This tag specifies the time as seconds since the epoch at which the package was built.
RPMTAG_BUILDHOST

This tag specifies the hostname of the system 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 associated 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.

22.2.5 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 22-16 CPIO File Format

<table>
<thead>
<tr>
<th>CPIO Header</th>
<th>Header structure as defined below.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>NUL terminated ASCII string containing the name of the file.</td>
</tr>
<tr>
<td>Padding</td>
<td>0-3 bytes as needed to align the file stream to a 4 byte boundary.</td>
</tr>
<tr>
<td>File data</td>
<td>The contents of the file.</td>
</tr>
<tr>
<td>Padding</td>
<td>0-3 bytes as needed to align the file stream to a 4 byte boundary.</td>
</tr>
</tbody>
</table>

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 the CPIO Header shall match the values contained in the Header Section.

```c
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.
22 Software Installation

22.1 File Signature

The signature is

```
end of file
```

22.2 File Checksum

The checksum is

```
end of file
```

22.3 Package Script Restrictions

Scripts used as part of the package install and uninstall shall only use commands and interfaces that are specified by the LSB. All other commands are not guaranteed to be present, or to behave in expected ways.

Packages shall not use RPM triggers.

Packages shall not depend on the order in which scripts are executed (pre-install, pre-uninstall, etc), when doing an upgrade.
22.4 Package Tools

The LSB does not specify the interface to the tools used to manipulate LSB-conformant packages. Each conforming implementation shall provide documentation for installing LSB packages.

22.5 Package Naming

Packages supplied by implementations and applications shall follow the following rules for the name field within the package. These rules are not required for the filename of the package file itself.

Note: There are discrepancies among implementations 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.

The following rules apply to the name field alone, not including any release or version.

Note: 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.

• If the name begins with lsb- and contains no other hyphens, the name shall be assigned by the Linux Assigned Names and Numbers Authority (http://www.lanana.org) (LANANA), which shall maintain a registry of LSB names. The name may be registered by either an implementation or an application.

• If the package name begins with lsb- and contains more than one hyphen (for example lsb-distro.example.com-database or lsb-gnome-gnumeric), then the portion of the package name between first and second hyphens shall either be an LSB provider name assigned by the LANANA, or it may be one of the 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 an implementation or an application.

• Package names containing no hyphens are reserved for use by implementations. Applications shall not use such names.

• Package names which do not start with lsb- and which contain a hyphen are open to both implementations and applications. Implementations 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.

Note: Widespread existing practice includes such names as ssh-common, ssh-client, kernel-pcmcia, and the like. Possible alternative names include sshcommon, foolinux-ssssh-common (where foolinux is registered to the implementation), or lsb-foolinux-ssh-common.

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.

Note: If an application vendor has domain name such as visicalc.example.com and has registered visicalc as a provider name, they might name packages visicalc-base, visicalc.example.com-charting, and the like.
Package names in this namespace are available to both the implementation and an application. Implementations 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-`).

### 22.6 Package Dependencies

Packages shall have a dependency that indicates which LSB modules are required. LSB module descriptions are dash separated tuples containing the name 'lsb', the module name, and the architecture name. The following dependencies may be used.

- **lsb-core-arch**
  
  This dependency is used to indicate that the application is dependent on features contained in the LSB-Core specification.

- **lsb-core-noarch**
  
  This dependency is used to indicate that the application is dependent on features contained in the LSB-Core specification and that the package does not contain any architecture specific files.

These dependencies shall have a version of 3.0.

Packages shall not depend on other system-provided dependencies. They shall not depend on non-system-provided dependencies unless the package provider also makes available the LSB conforming packages needed to satisfy such dependencies.

Other modules in the LSB may supplement this list. The architecture specific dependencies are described in the relevant architecture specific LSB.

### 22.7 Package Architecture Considerations

Packages which do not contain any architecture specific files should specify an architecture of `noarch`. An LSB runtime environment shall accept values `noarch`, or the value specified in the relevant architecture specific part of ISO/IEC 23360.

Additional specifications or restrictions may be found in the architecture specific LSB specification.
Annex A Alphabetical Listing of Interfaces

A.1 libc

The behavior of the interfaces in this library is specified by the following Standards.

Large File Support [LFS]
This Specification [LSB]
SUSv2 [SUSv2]
ISO POSIX (2003) [SUSv3]
SVID Issue 3 [SVID.3]
SVID Issue 4 [SVID.4]

Table A-1 libc Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Standards</th>
<th>Standards</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IO_feof[LSB]</td>
<td>getservbyname[SUSv3]</td>
<td>setstate[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>_IO_getc[LSB]</td>
<td>getservbyport[SUSv3]</td>
<td>setuid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>_IO_putc[LSB]</td>
<td>getserver[SUSv3]</td>
<td>setutent[LSB]</td>
<td></td>
</tr>
<tr>
<td>_IO_puts[LSB]</td>
<td>getsid[SUSv3]</td>
<td>setuxtent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__assert_fail[LSB]</td>
<td>getsockname[SUSv3]</td>
<td>setvbuf[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__ctype_b_loc(GLIBC_2.3)[LSB]</td>
<td>getsockopt[LSB]</td>
<td>shmat[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__ctype_get_mb_cur_max[LSB]</td>
<td>getsubopt[SUSv3]</td>
<td>shmtct[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__ctype_tolower_loc(GLIBC_2.3)[LSB]</td>
<td>gettext[LSB]</td>
<td>shmdt[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__ctype_toupper_loc(GLIBC_2.3)[LSB]</td>
<td>gettimeofday[SUSv3]</td>
<td>shmget[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__cxa_atexit[LSB]</td>
<td>getuid[SUSv3]</td>
<td>shutdown[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__cxa_finalize[LSB]</td>
<td>getutent[LSB]</td>
<td>sigaction[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__errno_location[LSB]</td>
<td>getutent_r[LSB]</td>
<td>sigaddset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fxstat[LSB]</td>
<td>getutxent[SUSv3]</td>
<td>sigaltstack[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fxstat64[LSB]</td>
<td>getutxid[SUSv3]</td>
<td>sig╙set[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__getpagesize[LSB]</td>
<td>getw[SUSv2]</td>
<td>sigemptyset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__getpgid[LSB]</td>
<td>getwc[SUSv3]</td>
<td>sigfillset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__h_errno_location[LSB]</td>
<td>getwchar[SUSv3]</td>
<td>sighold[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isinff[LSB]</td>
<td>getwd[SUSv3]</td>
<td>sigignore[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isinf[LSB]</td>
<td>glob[SUSv3]</td>
<td>siginterrupt[SUSv3]</td>
<td></td>
</tr>
</tbody>
</table>

© 2007 Linux Foundation
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__isinfl[LSB]</td>
<td>glob64[LSB]</td>
</tr>
<tr>
<td>__isnan[LSB]</td>
<td>globfree[SUSv3]</td>
</tr>
<tr>
<td>__isnanf[LSB]</td>
<td>globfree64[LSB]</td>
</tr>
<tr>
<td>__isnanl[LSB]</td>
<td>gmtime[SUSv3]</td>
</tr>
<tr>
<td>__libc_current_sigrtmax[LSB]</td>
<td>gmtime_r[SUSv3]</td>
</tr>
<tr>
<td>__libc_current_sigrtmin[LSB]</td>
<td>grantpt[SUSv3]</td>
</tr>
<tr>
<td>__libc_start_main[LSB]</td>
<td>hcreate[SUSv3]</td>
</tr>
<tr>
<td>__lxstat[LSB]</td>
<td>hdestroy[SUSv3]</td>
</tr>
<tr>
<td>__lxstat64[LSB]</td>
<td>hsearch[SUSv3]</td>
</tr>
<tr>
<td>__mempcpy[LSB]</td>
<td>htonl[SUSv3]</td>
</tr>
<tr>
<td>__rawmemchr[LSB]</td>
<td>htons[SUSv3]</td>
</tr>
<tr>
<td>__register_atfork(GLIBC_2.3.2)[LSB]</td>
<td>iconv[SUSv3]</td>
</tr>
<tr>
<td>__sigsetjmp[LSB]</td>
<td>iconv_close[SUSv3]</td>
</tr>
<tr>
<td>__stpcpy[LSB]</td>
<td>if_freenameindex[SUSv3]</td>
</tr>
<tr>
<td>__strtod_internal[LSB]</td>
<td>if_indextoname[SUSv3]</td>
</tr>
<tr>
<td>__strtof_internal[LSB]</td>
<td>if_nameindex[SUSv3]</td>
</tr>
<tr>
<td>__strtok_r[LSB]</td>
<td>if_nametoindex[SUSv3]</td>
</tr>
<tr>
<td>__strtol_internal[LSB]</td>
<td>imaxabs[SUSv3]</td>
</tr>
<tr>
<td>__strtold_internal[LSB]</td>
<td>imaxdiv[SUSv3]</td>
</tr>
<tr>
<td>__strtoll_internal[LSB]</td>
<td>index[SUSv3]</td>
</tr>
<tr>
<td>__strtoul_internal[LSB]</td>
<td>inet_addr[SUSv3]</td>
</tr>
<tr>
<td>__stroull_internal[LSB]</td>
<td>inet_aton[LSB]</td>
</tr>
<tr>
<td>__sysconf[LSB]</td>
<td>inet_ntoa[SUSv3]</td>
</tr>
<tr>
<td>__sysv_signal[LSB]</td>
<td>inet_ntop[SUSv3]</td>
</tr>
<tr>
<td>__wcstod_internal[LSB]</td>
<td>inet_pton[SUSv3]</td>
</tr>
<tr>
<td>__wcstof_internal[LSB]</td>
<td>initgroups[SUSv3]</td>
</tr>
<tr>
<td>__wcstol_internal[LSB]</td>
<td>initstate[SUSv3]</td>
</tr>
<tr>
<td>__wcstoul_internal[LSB]</td>
<td>insq[SUSv3]</td>
</tr>
<tr>
<td>__xmknod[LSB]</td>
<td>ioctl[LSB]</td>
</tr>
<tr>
<td>__xstrlchr[LSB]</td>
<td>isalnum[SUSv3]</td>
</tr>
<tr>
<td>__xstrlstr[LSB]</td>
<td>isdigit[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>__xpg_basename[LSB]</td>
<td>isalpha[SUSv3] stpcpy[LSB]</td>
</tr>
<tr>
<td>__xpg_strerror_r(GLIBC_2.3.4)[LSB]</td>
<td>isatty[SUSv3] strcasecmp[SUSv3]</td>
</tr>
<tr>
<td>__xstat[LSB]</td>
<td>isblank[SUSv3] strcasestr[SUSv3]</td>
</tr>
<tr>
<td>__xstat64[LSB]</td>
<td>iscntrl[SUSv3] stcat[SUSv3]</td>
</tr>
<tr>
<td>a64l[SUSv3]</td>
<td>isspace[SUSv3] strndup[SUSv3]</td>
</tr>
<tr>
<td>abs[SUSv3]</td>
<td>iswalnum[SUSv3] strerror_r[LSB]</td>
</tr>
<tr>
<td>access[SUSv3]</td>
<td>iswblank[SUSv3] strftime[SUSv3]</td>
</tr>
<tr>
<td>acct[LSB]</td>
<td>iswcntl[SUSv3] strlen[SUSv3]</td>
</tr>
<tr>
<td>adjtime[LSB]</td>
<td>iswctype[SUSv3] strncasecmp[SUSv3]</td>
</tr>
<tr>
<td>asctime_r[SUSv3]</td>
<td>iswlower[SUSv3] strncpy[SUSv3]</td>
</tr>
<tr>
<td>asprintf[LSB]</td>
<td>iswprint[SUSv3] strndup[LSB]</td>
</tr>
<tr>
<td>basename[LSB]</td>
<td>jrand48[SUSv3] strsignal[LSB]</td>
</tr>
<tr>
<td>bcmp[SUSv3]</td>
<td>key_decryptsession[SVID.D.3] strspn[SUSv3]</td>
</tr>
<tr>
<td>bcopy[SUSv3]</td>
<td>kill[LSB] strstr[SUSv3]</td>
</tr>
<tr>
<td>bind_textdomain_codeset[LSB]</td>
<td>l64a[SUSv3] strtol[SUSv3]</td>
</tr>
<tr>
<td>bindresvport[LSB]</td>
<td>labs[SUSv3] strtoimax[SUSv3]</td>
</tr>
<tr>
<td>bindtextdomain[LSB]</td>
<td>lchown[SUSv3]</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>brk[SUSv2]</td>
<td>lcong48[SUSv3]</td>
</tr>
<tr>
<td>bsearch[SUSv3]</td>
<td>lfind[SUSv3]</td>
</tr>
<tr>
<td>btowc[SUSv3]</td>
<td>link[LSB]</td>
</tr>
<tr>
<td>catgets[SUSv3]</td>
<td>localeconv[SUSv3]</td>
</tr>
<tr>
<td>cfgetispeed[SUSv3]</td>
<td>localtime_r[SUSv3]</td>
</tr>
<tr>
<td>cfgetospeed[SUSv3]</td>
<td>lockf[SUSv3]</td>
</tr>
<tr>
<td>cfmakeraw[LSB]</td>
<td>lockf64[LFS]</td>
</tr>
<tr>
<td>cfsetspeed[LSB]</td>
<td>lsearch[SUSv3]</td>
</tr>
<tr>
<td>chdir[SUSv3]</td>
<td>lseek[SUSv3]</td>
</tr>
<tr>
<td>chmod[SUSv3]</td>
<td>makecontext[SUSv3]</td>
</tr>
<tr>
<td>chown[SUSv3]</td>
<td>malloc[SUSv3]</td>
</tr>
<tr>
<td>chroot[SUSv2]</td>
<td>mblen[SUSv3]</td>
</tr>
<tr>
<td>clearerr[SUSv3]</td>
<td>mbrlen[SUSv3]</td>
</tr>
<tr>
<td>clnt_create[SVID.4]</td>
<td>mbtowc[SUSv3]</td>
</tr>
<tr>
<td>clock[SUSv3]</td>
<td>memchr[SUSv3]</td>
</tr>
<tr>
<td>close[SUSv3]</td>
<td>memcmp[SUSv3]</td>
</tr>
<tr>
<td>closedir[SUSv3]</td>
<td>memcpy[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>closelog</td>
<td>SUSv3</td>
</tr>
<tr>
<td>connect</td>
<td>SUSv3</td>
</tr>
<tr>
<td>creat</td>
<td>SUSv3</td>
</tr>
<tr>
<td>creat64</td>
<td>LSB</td>
</tr>
<tr>
<td>ctermid</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ctime</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ctime_r</td>
<td>SUSv3</td>
</tr>
<tr>
<td>cuserid</td>
<td>SUSv2</td>
</tr>
<tr>
<td>daemon</td>
<td>LSB</td>
</tr>
<tr>
<td>dcgettext</td>
<td>LSB</td>
</tr>
<tr>
<td>dcngettext</td>
<td>LSB</td>
</tr>
<tr>
<td>dgetext</td>
<td>LSB</td>
</tr>
<tr>
<td>difftime</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dirname</td>
<td>SUSv3</td>
</tr>
<tr>
<td>div</td>
<td>SUSv3</td>
</tr>
<tr>
<td>drand48</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dup</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dup2</td>
<td>SUSv3</td>
</tr>
<tr>
<td>duplocale</td>
<td>GLIBC_2.3</td>
</tr>
<tr>
<td>ecvt</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endgrent</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endprotoent</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endpwent</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endservent</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endutent</td>
<td>LSB</td>
</tr>
<tr>
<td>endutxent</td>
<td>SUSv3</td>
</tr>
<tr>
<td>erand48</td>
<td>SUSv3</td>
</tr>
<tr>
<td>err</td>
<td>LSB</td>
</tr>
<tr>
<td>error</td>
<td>LSB</td>
</tr>
<tr>
<td>errx</td>
<td>LSB</td>
</tr>
<tr>
<td>execl</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>execle[SUSv3]</td>
<td>ntohl[SUSv3]</td>
</tr>
<tr>
<td>execvp[SUSv3]</td>
<td>openlog[SUSv3]</td>
</tr>
<tr>
<td>fclose[SUSv3]</td>
<td>pipe[SUSv3]</td>
</tr>
<tr>
<td>fclose_unlocked[LSB]</td>
<td>pmap_getport[LSB]</td>
</tr>
<tr>
<td>fclose_unlocked[LSB]</td>
<td>pmap_set[LSB]</td>
</tr>
<tr>
<td>fdopen[SUSv3]</td>
<td>pmap_unset[LSB]</td>
</tr>
<tr>
<td>flush[SUSv3]</td>
<td>posix_fadvise64[LSB]</td>
</tr>
<tr>
<td>flush_unlocked[LSB]</td>
<td>posix_fallocate[SUSv3]</td>
</tr>
<tr>
<td>ffs[SUSv3]</td>
<td>posix_fallocate64[LSB]</td>
</tr>
<tr>
<td>fgetpos64[LFS]</td>
<td>posix_openpt[SUSv3]</td>
</tr>
<tr>
<td>flock[LFB]</td>
<td>posixspawn_file_actions_init[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>fopen64[LFS]</td>
<td>posix_spawnattr_getschedpolicy[SUSv3]</td>
</tr>
<tr>
<td>freenode(Glibc_2.3)[LSB]</td>
<td>printf[SUSv3]</td>
</tr>
<tr>
<td>freopen64[LFS]</td>
<td>psignal[LSB]</td>
</tr>
<tr>
<td>fscanf[LSB]</td>
<td>ptsname[SUSv3]</td>
</tr>
<tr>
<td>fseeko64[LFS]</td>
<td>putchar[SUSv3]</td>
</tr>
<tr>
<td>fsetpos64[LFS]</td>
<td>putenv[SUSv3]</td>
</tr>
<tr>
<td>fstatfs[LSB]</td>
<td>puts[SUSv3]</td>
</tr>
<tr>
<td>fstatfs64[LSB]</td>
<td>pututxline[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Source</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
</tr>
<tr>
<td>fclose</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fclose64</td>
<td>LFS</td>
</tr>
<tr>
<td>fsync</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_fseek</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_fseek64</td>
<td>LFS</td>
</tr>
<tr>
<td>_ftime</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_ftime64</td>
<td>LFS</td>
</tr>
<tr>
<td>_ftell</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_ftell64</td>
<td>LFS</td>
</tr>
<tr>
<td>_fwrite</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_fwrite64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getbackground</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getcontext</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getdata</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getegid</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_geteuid</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getenv</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getenv64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getgid</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_gethostname</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getимв</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getpagesize</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getpagesize64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getpos</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getres</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getsockopt</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getsockopt64</td>
<td>LFS</td>
</tr>
<tr>
<td>_gettimeofday</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_gettimeofday64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getuid</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_getpagesize64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getpagesize64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>_getres64</td>
<td>LFS</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>getgrent[SUSv3]</td>
<td>sched_get_priority_min</td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getgrgid[SUSv3]</td>
<td>sched_getparam[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getgrgid_r[SUSv3]</td>
<td>sched_getscheduler[SUSv3]</td>
</tr>
<tr>
<td>getgrnam[SUSv3]</td>
<td>sched_rr_get_interval[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>wprintf[SUSv3]</td>
</tr>
<tr>
<td>getgrnam_r[SUSv3]</td>
<td>sched_setparam[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>write[SUSv3]</td>
</tr>
<tr>
<td>getgrouplist[LSB]</td>
<td>sched_setscheduler[LSB]</td>
</tr>
<tr>
<td></td>
<td>wrintv[SUSv3]</td>
</tr>
<tr>
<td>getgroups[SUSv3]</td>
<td>sched_yield[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>wscanf[LSB]</td>
</tr>
<tr>
<td>gethostbyaddr[SUSv3]</td>
<td>seed48[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_accepted_reply[SVID D.3]</td>
</tr>
<tr>
<td>gethostbyaddr_r[LSB]</td>
<td>seekdir[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_array[SVID D.3]</td>
</tr>
<tr>
<td>gethostbyname[SUSv3]</td>
<td>select[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_bool[SVID D.3]</td>
</tr>
<tr>
<td>gethostbyname2[LSB]</td>
<td>semctl[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_bytes[SVID D.3]</td>
</tr>
<tr>
<td>gethostbyname2_r[LSB]</td>
<td>semget[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_callhdr[SVID D.3]</td>
</tr>
<tr>
<td>gethostbyname_r[LSB]</td>
<td>semop[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_callmsg[SVID D.3]</td>
</tr>
<tr>
<td>gethostid[SUSv3]</td>
<td>send[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_char[SVID D.3]</td>
</tr>
<tr>
<td>gethostname[SUSv3]</td>
<td>sendmsg[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_double[SVID D.3]</td>
</tr>
<tr>
<td>getitimer[SUSv3]</td>
<td>sendto[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_enum[SVID D.3]</td>
</tr>
<tr>
<td>getloadavg[LSB]</td>
<td>setbuf[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_float[SVID D.3]</td>
</tr>
<tr>
<td>getlogin[SUSv3]</td>
<td>setbuffer[LSB]</td>
</tr>
<tr>
<td></td>
<td>xdr_free[SVID D.3]</td>
</tr>
<tr>
<td>getlogin_r[SUSv3]</td>
<td>setcontext[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_int[SVID D.3]</td>
</tr>
<tr>
<td>getnameinfo[SUSv3]</td>
<td>setegid[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_long[SVID D.3]</td>
</tr>
<tr>
<td>getopt[LSB]</td>
<td>setenv[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdrOpaque[SVID D.3]</td>
</tr>
<tr>
<td>getopt_long[LSB]</td>
<td>seteuid[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdrOpaque_auth[SVID D.3]</td>
</tr>
<tr>
<td>getopt_long_only[LSB]</td>
<td>setgid[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_pointer[SVID D.3]</td>
</tr>
<tr>
<td>getpagesize[LSB]</td>
<td>setgrent[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_reference[SVID D.3]</td>
</tr>
<tr>
<td>getpeername[SUSv3]</td>
<td>setgroups[LSB]</td>
</tr>
<tr>
<td></td>
<td>xdr_rejected_reply[SVID D.3]</td>
</tr>
<tr>
<td>getpgid[SUSv3]</td>
<td>sethostname[LSB]</td>
</tr>
<tr>
<td></td>
<td>xdr_replymsg[SVID D.3]</td>
</tr>
<tr>
<td>getpgpr[SUSv3]</td>
<td>setitimer[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_short[SVID D.3]</td>
</tr>
<tr>
<td>getpid[SUSv3]</td>
<td>setlocale[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>xdr_string[SVID D.3]</td>
</tr>
</tbody>
</table>
### A.2 libcrypt

The behavior of the interfaces in this library is specified by the following Standards.

**ISO POSIX (2003) [SUSv3]**

### Table A-3 libcrypt Function Interfaces

|--------------|----------------|---------------|

### A.3 libdl

The behavior of the interfaces in this library is specified by the following Standards.

**This Specification [LSB]**

**ISO POSIX (2003) [SUSv3]**

### Table A-4 libdl Function Interfaces

<table>
<thead>
<tr>
<th>dladdr[LSB]</th>
<th>dlerror[SUSv3]</th>
<th>dlsym[LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlclose[SUSv3]</td>
<td>dlerror[LSB]</td>
<td></td>
</tr>
</tbody>
</table>
**A.4 libm**

The behavior of the interfaces in this library is specified by the following Standards.

ISO C (1999) [ISOC99]
This Specification [LSB]
ISO POSIX (2003) [SUSv3]
SVID Issue 3 [SVID.3]

Table A-5 libm Function Interfaces

<table>
<thead>
<tr>
<th>Function (LIB)</th>
<th>Function (SUSv3)</th>
<th>Function (SUSv3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>__finite[LSB]</td>
<td>csinl[SUSv3]</td>
<td>llroundf[SUSv3]</td>
</tr>
<tr>
<td>__finitel[LSB]</td>
<td>csqrt[SUSv3]</td>
<td>llroundl[SUSv3]</td>
</tr>
<tr>
<td>__finitel[LSB]</td>
<td>csqrtl[SUSv3]</td>
<td>log[SUSv3]</td>
</tr>
<tr>
<td>__fpclassify[LSB]</td>
<td>csqrtl[SUSv3]</td>
<td>log10[SUSv3]</td>
</tr>
<tr>
<td>__fpclassifyf[LSB]</td>
<td>ctan[SUSv3]</td>
<td>log10f[SUSv3]</td>
</tr>
<tr>
<td>__signbit[LSB]</td>
<td>ctanl[SUSv3]</td>
<td>log10l[SUSv3]</td>
</tr>
<tr>
<td>__signbitl[LSB]</td>
<td>ctanh[SUSv3]</td>
<td>log2[SUSv3]</td>
</tr>
<tr>
<td>acos[SUSv3]</td>
<td>ctanhf[SUSv3]</td>
<td>log1p[SUSv3]</td>
</tr>
<tr>
<td>acosf[SUSv3]</td>
<td>ctanhlf[SUSv3]</td>
<td>log1pf[SUSv3]</td>
</tr>
<tr>
<td>acosh[SUSv3]</td>
<td>ctanlh[SUSv3]</td>
<td>log1pl[SUSv3]</td>
</tr>
<tr>
<td>acosl[SUSv3]</td>
<td>dremf[LSB]</td>
<td>log2f[SUSv3]</td>
</tr>
<tr>
<td>acosl[SUSv3]</td>
<td>dremf[LSB]</td>
<td>log2l[SUSv3]</td>
</tr>
<tr>
<td>asinf[SUSv3]</td>
<td>erfc[SUSv3]</td>
<td>logbl[SUSv3]</td>
</tr>
<tr>
<td>atan2[SUSv3]</td>
<td>exp10f[LSB]</td>
<td>lroundf[SUSv3]</td>
</tr>
<tr>
<td>atan2f[SUSv3]</td>
<td>exp10f[LSB]</td>
<td>lroundf[SUSv3]</td>
</tr>
<tr>
<td>atan2l[SUSv3]</td>
<td>exp10l[LSB]</td>
<td>lroundl[SUSv3]</td>
</tr>
<tr>
<td>atanhf[SUSv3]</td>
<td>exp2[SUSv3]</td>
<td>matherr[SVID.3]</td>
</tr>
<tr>
<td>atanh[SUSv3]</td>
<td>exp2f[SUSv3]</td>
<td>modf[SUSv3]</td>
</tr>
<tr>
<td>atanh[SUSv3]</td>
<td>expl[SUSv3]</td>
<td>modfl[SUSv3]</td>
</tr>
<tr>
<td>atanl[SUSv3]</td>
<td>expm1[SUSv3]</td>
<td>nan[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>SUSv3</td>
<td>Function</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>------------------</td>
</tr>
<tr>
<td>cabs[SUSv3]</td>
<td>expm1f[SUSv3]</td>
<td>nanf[SUSv3]</td>
</tr>
<tr>
<td>cabsf[SUSv3]</td>
<td>expm1l[SUSv3]</td>
<td>nanl[SUSv3]</td>
</tr>
<tr>
<td>cacosl[SUSv3]</td>
<td>fdim[SUSv3]</td>
<td>nextafter[SUSv3]</td>
</tr>
<tr>
<td>cacoshf[SUSv3]</td>
<td>fedisableexcept[LSB]</td>
<td>nexttowardf[SUSv3]</td>
</tr>
<tr>
<td>cacoshl[SUSv3]</td>
<td>feenableexcept[LSB]</td>
<td>nexttowardl[SUSv3]</td>
</tr>
<tr>
<td>casinf[SUSv3]</td>
<td>fegetexceptflag[SUSv3]</td>
<td>pow10[LSEB]</td>
</tr>
<tr>
<td>casinh[SUSv3]</td>
<td>fegetround[SUSv3]</td>
<td>pow10[LSB]</td>
</tr>
<tr>
<td>catan[SUSv3]</td>
<td>fetexceptflag[SUSv3]</td>
<td>remainderf[SUSv3]</td>
</tr>
<tr>
<td>catanhl[SUSv3]</td>
<td>finite[LSB]</td>
<td>remquol[SUSv3]</td>
</tr>
<tr>
<td>catanl[SUSv3]</td>
<td>finitef[LSB]</td>
<td>rint[SUSv3]</td>
</tr>
<tr>
<td>cbtf[SUSv3]</td>
<td>fintel[LSB]</td>
<td>rintf[SUSv3]</td>
</tr>
<tr>
<td>cbtr[SUSv3]</td>
<td>floorf[SUSv3]</td>
<td>round[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>clog10[LSB]</td>
<td>gamma[LSB]</td>
<td>sincosf[LSB]</td>
</tr>
<tr>
<td>clog10f[LSB]</td>
<td>gammaf[LSB]</td>
<td>sincosi[LSB]</td>
</tr>
<tr>
<td>clog10l[LSB]</td>
<td>gammal[LSB]</td>
<td>sinf[SUSv3]</td>
</tr>
<tr>
<td>copysignf[SUSv3]</td>
<td>j0[SUSv3]</td>
<td>sqrtl[SUSv3]</td>
</tr>
<tr>
<td>copysignl[SUSv3]</td>
<td>j0l[LSB]</td>
<td>tan[SUSv3]</td>
</tr>
<tr>
<td>cos[SUSv3]</td>
<td>j0l[LSB]</td>
<td>tan[SUSv3]</td>
</tr>
<tr>
<td>cosf[SUSv3]</td>
<td>j1[SUSv3]</td>
<td>tanb[SUSv3]</td>
</tr>
<tr>
<td>cosh[SUSv3]</td>
<td>j1l[LSB]</td>
<td>tanhf[SUSv3]</td>
</tr>
<tr>
<td>coshf[SUSv3]</td>
<td>j1l[LSB]</td>
<td>tanhl[SUSv3]</td>
</tr>
<tr>
<td>coshl[SUSv3]</td>
<td>jnl[LSB]</td>
<td>tanl[SUSv3]</td>
</tr>
<tr>
<td>cosl[SUSv3]</td>
<td>jnl[LSB]</td>
<td>tgamma[SUSv3]</td>
</tr>
<tr>
<td>cpow[SUSv3]</td>
<td>jnl[LSB]</td>
<td>tgammap[SUSv3]</td>
</tr>
<tr>
<td>cprojl[SUSv3]</td>
<td>lgamma_r[LSB]</td>
<td>y0[SUSv3]</td>
</tr>
<tr>
<td>crealf[SUSv3]</td>
<td>lgammaf[SUSv3]</td>
<td>y0[LSB]</td>
</tr>
<tr>
<td>crealf[SUSv3]</td>
<td>lgammaf_r[LSB]</td>
<td>y0[LSB]</td>
</tr>
</tbody>
</table>


© 2007 Linux Foundation
creall[SUSv3]  lgammal[SUSv3]  y1[SUSv3]
csin[SUSv3]  lgammal_r[LSB]  y1f[LSB]
csinf[SUSv3]  llrint[SUSv3]  y1lf[LSB]
csinhf[SUSv3]  llrintl[SUSv3]  ynf[LSB]
csinhl[SUSv3]  llround[SUSv3]  ynl[LSB]

Table A-6 libm Data Interfaces

signgam[SUSv3]

A.5 libncurses

The behavior of the interfaces in this library is specified by the following Standards.
X/Open Curses [SUS-CURSES]

Table A-7 libncurses Function Interfaces

<table>
<thead>
<tr>
<th>addch[SUS-CURSES]</th>
<th>mvdelch[SUS-CURSES]</th>
<th>slk_refresh[SUS-CURSES]</th>
</tr>
</thead>
<tbody>
<tr>
<td>addchnstr[SUS-CURSES]</td>
<td>mvderwin[SUS-CURSES]</td>
<td>slk_restore[SUS-CURSES]</td>
</tr>
<tr>
<td>addnstr[SUS-CURSES]</td>
<td>mvgetnstr[SUS-CURSES]</td>
<td>slk_touch[SUS-CURSES]</td>
</tr>
<tr>
<td>attr_get[SUS-CURSES]</td>
<td>mvhline[SUS-CURSES]</td>
<td>standout[SUS-CURSES]</td>
</tr>
<tr>
<td>attr_off[SUS-CURSES]</td>
<td>mvinch[SUS-CURSES]</td>
<td>start_color[SUS-CURSES]</td>
</tr>
<tr>
<td>attr_on[SUS-CURSES]</td>
<td>mvinchnstr[SUS-CURSES]</td>
<td>subpad[SUS-CURSES]</td>
</tr>
<tr>
<td>attrset[SUS-CURSES]</td>
<td>mvinsnstr[SUS-CURSES]</td>
<td>termname[SUS-CURSES]</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>can_change_color[SUS-CURSES]</td>
<td>mwwaddchnstr[SUS-CURSES]</td>
<td>tigetnum[SUS-CURSES]</td>
</tr>
<tr>
<td>clear[SUS-CURSES]</td>
<td>mwwaddstr[SUS-CURSES]</td>
<td>touchline[SUS-CURSES]</td>
</tr>
<tr>
<td>color_content[SUS-CURSES]</td>
<td>mwwgetnstr[SUS-CURSES]</td>
<td>typeahead[SUS-CURSES]</td>
</tr>
<tr>
<td>delwin[SUS-CURSES]</td>
<td>mwwinsnstr[SUS-</td>
<td>vwsscanw[SUS-CURSES]</td>
</tr>
<tr>
<td>Function</td>
<td>CURSES</td>
<td>CURSES</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>derwin</td>
<td>mvwprintw</td>
<td>waddch</td>
</tr>
<tr>
<td>doupdate</td>
<td>mvwscanw</td>
<td>waddchnstr</td>
</tr>
<tr>
<td>dupwin</td>
<td>mvwvline</td>
<td>waddchstr</td>
</tr>
<tr>
<td>echo</td>
<td>napms</td>
<td>waddnstr</td>
</tr>
<tr>
<td>echochar</td>
<td>newpad</td>
<td>waddstr</td>
</tr>
<tr>
<td>endwin</td>
<td>newterm</td>
<td>wattr_get</td>
</tr>
<tr>
<td>erase</td>
<td>newwin</td>
<td>wattr_off</td>
</tr>
<tr>
<td>erasechar</td>
<td>nl</td>
<td>wattr_on</td>
</tr>
<tr>
<td>filter</td>
<td>nocbreak</td>
<td>wattr_set</td>
</tr>
<tr>
<td>flash</td>
<td>nodelay</td>
<td>wattroff</td>
</tr>
<tr>
<td>flushinp</td>
<td>noecho</td>
<td>watttron</td>
</tr>
<tr>
<td>getbkgd</td>
<td>nonl</td>
<td>wattset</td>
</tr>
<tr>
<td>getch</td>
<td>noqiflush</td>
<td>wbkgd</td>
</tr>
<tr>
<td>getnstr</td>
<td>noraw</td>
<td>wbkgdset</td>
</tr>
<tr>
<td>getstr</td>
<td>notimeout</td>
<td>wborder</td>
</tr>
<tr>
<td>getwin</td>
<td>overlay</td>
<td>wchgat</td>
</tr>
<tr>
<td>halfdelay</td>
<td>overwrite</td>
<td>wcmap</td>
</tr>
<tr>
<td>has_colors</td>
<td>pair_content</td>
<td>wclrtobot</td>
</tr>
<tr>
<td>has_ic</td>
<td>pechochar</td>
<td>wclrtœol</td>
</tr>
<tr>
<td>has_il</td>
<td>pnooutrefresh</td>
<td>wcolor_set</td>
</tr>
<tr>
<td>hline</td>
<td>prefresh</td>
<td>wcursyncup</td>
</tr>
<tr>
<td>idcok</td>
<td>printw</td>
<td>wdelch</td>
</tr>
<tr>
<td>idlok</td>
<td>putp</td>
<td>wdeleteLn</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>insnstr[SUS-CURSES]</td>
<td>scanw[SUS-CURSES]</td>
<td>winsdelln[SUS-CURSES]</td>
</tr>
<tr>
<td>is_linetouched[SUS-CURSES]</td>
<td>scr_set[SUS-CURSES]</td>
<td>winstr[SUS-CURSES]</td>
</tr>
<tr>
<td>is_wintouched[SUS-CURSES]</td>
<td>scr_restore[SUS-CURSES]</td>
<td>winsnstr[SUS-CURSES]</td>
</tr>
</tbody>
</table>
Table A-8 libncurses Data Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR_PAIRS[SUS-CURSES]</td>
<td>acs_map[SUS-CURSES]</td>
<td>stdscr[SUS-CURSES]</td>
</tr>
<tr>
<td>COLS[SUS-CURSES]</td>
<td>cur_term[SUS-CURSES]</td>
<td></td>
</tr>
</tbody>
</table>

A.6 libpam

The behavior of the interfaces in this library is specified by the following Standards.

This Specification [LSB]

Table A-9 libpam Function Interfaces

<table>
<thead>
<tr>
<th>pam_acct_mgmt[LSB]</th>
<th>pam_fail_delay[LSB]</th>
<th>pam_putenv[LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pam_authenticate[LSB]</td>
<td>pam_get_item[LSB]</td>
<td>pam_set_item[LSB]</td>
</tr>
<tr>
<td>pam_chauthtok[LSB]</td>
<td>pam_getenv[LSB]</td>
<td>pam_setcred[LSB]</td>
</tr>
<tr>
<td>pam_close_session[LSB]</td>
<td>pam_getenvlist[LSB]</td>
<td>pam_start[LSB]</td>
</tr>
<tr>
<td>pam_end[LSB]</td>
<td>pam_open_session[LSB]</td>
<td>pam_strerror[LSB]</td>
</tr>
</tbody>
</table>

A.7 libpthread

The behavior of the interfaces in this library is specified by the following Standards.

Large File Support [LFS]
This Specification [LSB]
ISO POSIX (2003) [SUSv3]
<table>
<thead>
<tr>
<th>Function</th>
<th>生产厂家</th>
<th>生产厂家</th>
<th>生产厂家</th>
</tr>
</thead>
</table>

© 2007 Linux Foundation
### A.8 librt

The behavior of the interfaces in this library is specified by the following Standards.

ISO POSIX (2003) [SUSv3]

### Table A-11 librt Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Standard</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock_getcpuclockid</td>
<td>SUSv3</td>
<td>clock_getres[GLIBC_2.3.4]</td>
</tr>
<tr>
<td>clock_getres</td>
<td>SUSv3</td>
<td>mq_open(GLIBC_2.3.4)[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setschedpolicy</td>
<td>SUSv3</td>
<td>pthread_attr_setschedpolicy</td>
</tr>
<tr>
<td>pthread_attr_setscope</td>
<td>SUSv3</td>
<td>pthread_attr_setscope</td>
</tr>
<tr>
<td>pthread_attr_setstack</td>
<td>SUSv3</td>
<td>pthread_attr_setstack</td>
</tr>
<tr>
<td>pthread_attr_setstackdr</td>
<td>SUSv3</td>
<td>pthread_attr_setstackdr</td>
</tr>
<tr>
<td>pthread_barrier_destroy</td>
<td>SUSv3</td>
<td>pthread_barrier_destroy</td>
</tr>
<tr>
<td>pthread_barrier_init</td>
<td>SUSv3</td>
<td>pthread_barrier_init</td>
</tr>
<tr>
<td>pthread_barrier_wait</td>
<td>SUSv3</td>
<td>pthread_barrier_wait</td>
</tr>
<tr>
<td>pthread_barrierattr_destroy</td>
<td>SUSv3</td>
<td>pthread_barrierattr_destroy</td>
</tr>
<tr>
<td>pthread_barrierattr_init</td>
<td>SUSv3</td>
<td>pthread_barrierattr_init</td>
</tr>
<tr>
<td>pthread_barrierattr_getpshared</td>
<td>SUSv3</td>
<td>pthread_barrierattr_getpshared</td>
</tr>
<tr>
<td>pthread_cancel</td>
<td>SUSv3</td>
<td>pthread_cancel</td>
</tr>
<tr>
<td>pthread_cond_broadcast</td>
<td>SUSv3</td>
<td>pthread_cond_broadcast</td>
</tr>
<tr>
<td>pthread_cond_destroy</td>
<td>SUSv3</td>
<td>pthread_cond_destroy</td>
</tr>
</tbody>
</table>
### Annex A Alphabetical Listing of Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock_gettime</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>clock_nanosleep</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>clock_settime</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>mq_send</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mq_setattr</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>timer_delete</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>timer_gettime</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>timer_getoverrun</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>timer_settime</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>timer_getoverrun</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>timer_settime</td>
<td>SUSv3</td>
<td></td>
</tr>
<tr>
<td>mq_close</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mq_timedreceive</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mq_timedsend</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mqunlink</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mq_unlink</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mq_notify</td>
<td>GLIBC_2.3.4</td>
<td>SUSv3</td>
</tr>
<tr>
<td>shm_open</td>
<td>SUSv3</td>
<td></td>
</tr>
</tbody>
</table>

### A.9 libutil

The behavior of the interfaces in this library is specified by the following Standards.

This Specification [LSB]

**Table A-12 libutil Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>forkpty</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>login</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>logout</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>openpt</td>
<td>LSB</td>
<td></td>
</tr>
</tbody>
</table>

### A.10 libz

The behavior of the interfaces in this library is specified by the following Standards.

This Specification [LSB]

**Table A-13 libz Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>adler32</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>compress</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>compress2</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>compressBound</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>crc32</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflate</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflateBound</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflateCopy</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflateEnd</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflateInit</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflateInit2</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzclose</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzerror</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzflush</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzgetc</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzgets</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gziprint</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzputc</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzputs</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzdopen</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzwrite</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>gzipopen</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflate</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflateEnd</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflateInit</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflateInit2</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflateSetDictionary</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflateSync</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>inflateSyncPoint</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>uncompress</td>
<td>LSB</td>
<td></td>
</tr>
<tr>
<td>deflateParams[LSB]</td>
<td>gzread[LSB]</td>
<td>zError[LSB]</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>deflateSetDictionary[LSB]</td>
<td>gzseek[LSB]</td>
<td></td>
</tr>
<tr>
<td>get_crc_table[LSB]</td>
<td>gzsetparams[LSB]</td>
<td></td>
</tr>
</tbody>
</table>
Annex B Future Directions (Informative)

B.1 Introduction

This appendix describes interfaces that are under development and aimed at future releases of this specification. At this stage, such interfaces are at best recommended practice, and do not constitute normative requirements of this specification. Applications may not assume that any system provides these interfaces.

We encourage system implementors and ISVs to provide these interfaces, and to provide feedback on their specification to lsb-spec@freestandards.org (mailto://lsb-spec@freestandards.org). These interfaces may well be further modified during the development process, and may be withdrawn if consensus cannot be reached.
B.2 Commands And Utilities

lsbinstall

Name

lsbinstall — installation tool for various types of data

Synopsis

/usr/lib/lsb/lsbinstall [-c | --check | -r | --remove] { -t type | --type=type } [-p package | --package=package] operand...

Description

The **lsbinstall** utility may be used to install certain types of files into system specific locations, repositories, or databases. This command may be used during a package post installation script to add package specific data to system wide repositories. A user may need appropriate privilege to invoke **lsbinstall**.

The operand (or operands) name an object of type *type* (see below) that belongs to a package named *package*. The combination of package name, object type and object name should be unique amongst all objects installed by **lsbinstall**. The **lsbinstall** utility may rename an object if another package already owns an object of the same type with the same name.

**Note**: If a namespace collision is detected by **lsbinstall**, it is unspecified how the object is renamed, although typical implementations may prepend the package name to the object in some way (e.g. *package.obj-name*). The **lsbinstall** utility may maintain a database of the mappings it has performed during installation in order to ensure that the correct object is removed during a subsequent removal operation.

Scripts installed by **lsbinstall** should not make use of the script name in order to decide on their functionality.

**Note**: It is appropriate for such a script to use the script name in error messages, usage statements, etc. The only guarantee made by **lsbinstall** is the effect that an installation (or removal) should have, not where a script is installed, or how it is named.

The **-p pkg** or **--package=pkg** is required for all object types unless explicitly noted below.

If the **-c** or **--check** option is specified, **lsbinstall** should test to see if there is an existing object of the type specified already installed. If there is, **lsbinstall** should print a message to its standard output and immediately exit with a status of zero. If there is no object of the type and name specified already installed, **lsbinstall** should exit with a non-zero status and take no further action.

If the **-r** or **--remove** is specified, the named object of the specified type should be removed or disabled from the system, except as noted below. The behavior is unspecified if the named object was not previously installed by **lsbinstall**.

**Note**: **lsbinstall** may rename objects during installation in order to prevent name collisions where another package has already installed an object with the given name. Using **lsbinstall --remove** will remove only the object belonging to the named package, and not the object belonging to another package.
Also note that the intent of the \texttt{--remove} option is to prevent the effect of the installed object; it should be sufficient to disable or comment out the addition in some way, while leaving the content behind. It is not intended that \texttt{--remove} be required to be the exact reverse of installation.

**Object Types**

The \texttt{-t type} or \texttt{--type=type} option should support at least the following types:

- **profile**
  
  install a profile script into a system specific location. There should be one operand, that names a profile shell script. The behavior is unspecified if this name does not have the suffix \texttt{.sh}.

  The \texttt{sh} utility should read and execute commands in its current execution environment from all such installed profile shell scripts when invoked as an interactive login shell, or if the \texttt{-i} (the letter \texttt{ell}) is specified (see Shell Invocation).

- **service**
  
  ensure a service name and number pair is known to the system service database. When installing, there must be at least two operands. The first operand should have the format \texttt{%-d/-s} with the port number and protocol values (e.g. \texttt{22/tcp}), and the second operand should be the name of the service. Any subsequent operands provide aliases for this service. The \texttt{-p pkg} or \texttt{--package=pkg} option is not required for service objects, and is ignored if specified. If any of the \texttt{-r}, \texttt{--remove}, \texttt{-c} or \texttt{--check} options are specified, there should be a single operand identifying the port and protocol values (with the same format as above).

  It should not be an error to attempt to add a service name to the system service database if that service name already exists for the same port and protocol combination. If the port and protocol combination was already present, but the name unknown, the name should be added as an alias to the existing entry. It should be an error to attempt to add a second entry for a given service name and protocol, but where the port number differs from an existing entry.

  If the \texttt{-r} or \texttt{--remove} is specified, the system service database need not be updated to remove or disable the named service.

- **inet**
  
  add an entry to the system's network super daemon configuration. If none of the \texttt{-r, --remove, --check} options are specified, the first operand should have the format:

  "\texttt{%-s/s:%s:%s:%s:%s:%s}"

  Otherwise, the first operand should have the format

  "\texttt{%-s/s}"  

  The fields in the first operand have the following meaning, in order:

  - \texttt{svc_name}
    
    The name of this service. If the name does not contain a \texttt{/}, this should match the name of an already installed service (see also \texttt{getservbyname()}). If the name contains a \texttt{/} character, the behavior is unspecified.
Rationale: This version of the LSB does not specify `getrpcbyname()` nor the existence or format of the `/etc/rpc` file. Therefore, installation of RPC based services is not specified at this point. A future version of this specification may require names containing a `/` character to be Remote Procedure Call based services.

`protocol`

The name of a protocol. The name should be one of those listed in `/etc/protocols`. If this attribute is not specified (i.e. a null value is passed), the system should use an implementation defined default protocol.

`socket_type`

One of the following values:

- `stream`
  
  the service will use a stream type socket.

- `dgram`
  
  the service will use a datagram type socket.

- `seqpacket`
  
  the service will use a sequenced packet type socket.

This field is not required for the `-c`, `--check`, `-r`, or `--remove` options.

`wait_flag`

If the value of this attribute is `wait`, once the service is started, no further requests for that service will be handled until the service exits. If the value is `nowait`, the network super daemon should continue to handle further requests for the given service while that service is running.

Note: If the service has the `socket_type` attribute set to `dgram`, the `wait_flag` attribute should be set to `wait`, since such services do not have any distinction between the socket used for listening and that used for accepting.

This field is not required for the `-c`, `--check`, `-r`, or `--remove` options.

`user[.group]`

The name of a user from the user login database, optionally followed by the name of a group from the group database. The service started to handle this request should run with the privileges of the specified user and group. This field is not required for the `-c`, `--check`, `-r`, or `--remove` options.

`server [arg ...]`

The name of a program to run to handle the request, optionally followed by any arguments required. The server name and each of its arguments is separated by whitespace. This field is not required for the `-c`, `--check`, `-r`, or `--remove` options.

If the implementation supports additional controls over services started through the inet super daemon, there may be additional, implementation-defined, operands.
**Rationale:** Systems that use the `xinetd` super daemon may support additional controls such as IP address restrictions, logging requirements, etc. The LSB does not require these additional controls. However, it was believed to be of sufficient benefit that implementations are granted permission to extend this interface as required.

**Examples**

```bash
lsbinstall --package=myapp --type=profile myco.com-prod.sh
```

Install the profile shell script for `myco.com-prod.sh`, part of the `myapp` package.

```bash
lsbinstall --package=myapp --check --type=profile myco.com-prod.sh
```

Test to see if the profile shell script for `myco.com-prod.sh`, as part of the `myapp` package, is installed correctly.

**Exit Status**

If the `-c` or `--check` option is specified, `lsbinstall` should exit with a zero status if an object of the specified type and name is already installed, or non-zero otherwise. Otherwise, `lsbinstall` should exit with a zero status if the object with the specified type and name was successfully installed (or removed if the `-r` or `--remove` option was specified), and non-zero if the installation (or removal) failed. On failure, a diagnostic message should be printed to the standard error file descriptor.
Annex C GNU Free Documentation License (Informative)

This specification is published under the terms of the GNU Free Documentation License, Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

C.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.

C.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.

C.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.

C.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.

C.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.

### C.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."

C.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.

C.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.

C.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.

C.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.

C.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.

C.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.