Linux Standard Base Core Specification 4.0
Linux Standard Base Core Specification 4.0
ISO/IEC 23360 Part 1:2008(E)
Copyright © 2008 Linux Foundation

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

Portions of the text may be copyrighted by the following parties:
- The Regents of the University of California
- Free Software Foundation
- Ian F. Darwin
- Paul Vixie
- BSDI (now Wind River)
- Andrew G Morgan
- Jean-loup Gailly and Mark Adler
- Massachusetts Institute of Technology
- Apple Inc.
- Easy Software Products
- artofcode LLC
- Till Kamppeter
- Manfred Wassman
- Python Software Foundation

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

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.
UNIX is a registered trademark of The Open Group.
LSB is a trademark of the Linux Foundation in the United States and other countries.
AMD is a trademark of Advanced Micro Devices, Inc.
Intel and Itanium are registered trademarks and Intel386 is a trademark of Intel Corporation.
PowerPC is a registered trademark and PowerPC Architecture is a trademark of the IBM Corporation.
S/390 is a registered trademark of the IBM Corporation.
OpenGL is a registered trademark of Silicon Graphics, Inc.
VII System Initialization

VI Execution Environment

V Commands and Utilities

IV Utility Libraries

III Interfaces and Implementations

II Software Development Environment

I System Overview

ISO/IEC 23360 Part 1:2008(E)

© 2008 Linux Foundation
VIII Users & Groups

21 Users & Groups

21.1 User and Group Database
21.2 User & Group Names
21.3 User ID Ranges
21.4 Rationale

IX Package Format and Installation

22 Software Installation

22.1 Introduction
22.2 Package File Format
22.3 Package Script Restrictions
22.4 Package Tools
22.5 Package Naming
22.6 Package Dependencies
22.7 Package Architecture Considerations

A Alphabetical Listing of Interfaces

A.1 libc
A.2 libcrypt
A.3 libdl
A.4 libm
A.5 libncurses
A.6 libpam
A.7 libpthread
A.8 librt
A.9 libutil
A.10 libz

B Future Directions (Informative)

B.1 Introduction
B.2 Commands And Utilities

lsbinstall

C GNU Free Documentation License (Informative)

C.1 PREAMBLE
C.2 APPLICABILITY AND DEFINITIONS
C.3 VERBATIM COPYING
C.4 COPYING IN QUANTITY
C.5 MODIFICATIONS
C.6 COMBINING DOCUMENTS
C.7 COLLECTIONS OF DOCUMENTS
C.8 AGGREGATION WITH INDEPENDENT WORKS
C.9 TRANSLATION
C.10 TERMINATION
C.11 FUTURE REVISIONS OF THIS LICENSE
C.12 How to use this License for your documents

© 2008 Linux Foundation
List of Figures

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

This is version 4.0 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 \texttt{x.y} or \texttt{x.y.z}. This version number carries the following meaning:

- The first number (\texttt{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 (\texttt{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 (\texttt{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>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filesystem Hierarchy</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><a href="http://www.unix.org/version3/">http://www.unix.org/version3/</a></td>
<td></td>
</tr>
</tbody>
</table>
## 2 References

<table>
<thead>
<tr>
<th>Itanium™ C++ ABI (Revision 1.83)</th>
<th><a href="http://refspecs.linux%C2%ADfoundation.org/cxxabi-1.83.html">http://refspecs.linux­foundation.org/cxxabi-1.83.html</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Large File Support</td>
<td><a href="http://www.UNIX%C2%ADsystems.org/version2/whatsnew/lfs20mar.html">http://www.UNIX­systems.org/version2/whatsnew/lfs20mar.html</a></td>
</tr>
<tr>
<td>POSIX 1003.1 2008</td>
<td><a href="http://www.unix.org/version4/">http://www.unix.org/version4/</a></td>
</tr>
<tr>
<td>SUSv2</td>
<td><a href="http://www.opengroup.org/publications/catalog/un.htm">http://www.opengroup.org/publications/catalog/un.htm</a></td>
</tr>
<tr>
<td>SVID Issue 3</td>
<td></td>
</tr>
<tr>
<td>SVID Issue 4</td>
<td><a href="http://www.caldera.com/developers/gabi41.pdf">http://www.caldera.com/developers/gabi41.pdf</a></td>
</tr>
<tr>
<td>System V ABI</td>
<td></td>
</tr>
<tr>
<td>X/Open Curses</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>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cairo API Reference</td>
<td>Cairo Vector Graphics API Specification for 1.0.2</td>
<td><a href="http://cairographics.org/manual-1.0.2">http://cairographics.org/manual-1.0.2</a></td>
</tr>
<tr>
<td>DWARF Debugging Information Format, Revision 2.0.0</td>
<td>DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)</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>DWARF Debugging Information Format, Revision 3.0.0 (Draft)</td>
<td>DWARF Debugging Information Format, Revision 3.0.0 (Draft)</td>
<td><a href="http://refspecs.linux-foundation.org/dwarf">http://refspecs.linux-foundation.org/dwarf</a></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>Mozilla's NSS SSL Reference</td>
<td>Mozilla's NSS SSL Reference</td>
<td><a href="http://www.mozilla.org/projects/security/pki/nss/ref/ssl/">http://www.mozilla.org/projects/security/pki/nss/ref/ssl/</a></td>
</tr>
<tr>
<td>PAM</td>
<td>Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar</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>Reference</td>
<td>Description</td>
<td>Reference</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</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...
3 Requirements

ISO/IEC 23360 Part 1:2008(E)

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.
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:** For symbols with versions which differ between architectures, the 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>
<thead>
<tr>
<th>Table 11-1 ELF Section Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>SHT_DYNAMIC</td>
</tr>
<tr>
<td>SHT_DYNNSYM</td>
</tr>
</tbody>
</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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>Section Code</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>SHT_NOTE</td>
<td>0x7</td>
<td>The section holds information that marks the file in some way. See <code>Note Section</code> in Chapter 5 of System V ABI Update for details.</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 <code>Initialization and Termination Functions</code> 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 <code>Relocation</code> 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...</td>
</tr>
</tbody>
</table>
files or type Elf64_Rela 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.

SHT_STRTAB 0x3  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.

SHT_SYMTAB 0x2  This section holds a symbol table. Currently, an object file may have either a section of SHT_SYMTAB type or a section of SHT_DYNSYM type, but not both. This restriction may be relaxed in the future. Typically, SHT_SYMTAB provides symbols for link editing, though it may also be used for dynamic linking. As a complete symbol table, it may contain many symbols unnecessary for dynamic linking.

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 11-3 ELF Special Sections

<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_EXECINST</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_EXECINST</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+SHF_STRINGS</td>
</tr>
<tr>
<td>.shstrtab</td>
<td>SHT_STRTAB</td>
<td>0</td>
</tr>
<tr>
<td>.strtab</td>
<td>SHT_STRTAB</td>
<td>SHF_ALLOC</td>
</tr>
</tbody>
</table>
### .symtab

<table>
<thead>
<tr>
<th>Section</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.symtab</td>
<td>SHT_SYMTAB</td>
<td>SHF_ALLOC</td>
</tr>
</tbody>
</table>

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.

### .tbss

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.

### .tbss

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.

.rodata1
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_ALLOCS 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_ALLOCS bit; otherwise, that bit will be off.

.tbs
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>Name</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ctors</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS+SHF_WRITE</td>
</tr>
<tr>
<td>.data.rel.ro</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS+SHF_WRITE</td>
</tr>
<tr>
<td>.dtors</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS+SHF_WRITE</td>
</tr>
<tr>
<td>.eh_frame</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS</td>
</tr>
<tr>
<td>.eh_frame_hdr</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS</td>
</tr>
<tr>
<td>.gcc_except_table</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS</td>
</tr>
<tr>
<td>.gnu.version</td>
<td>SHT_GNU_versym</td>
<td>SHF_ALLOCS</td>
</tr>
<tr>
<td>.gnu.version_d</td>
<td>SHT_GNU_veredef</td>
<td>SHF_ALLOCS</td>
</tr>
<tr>
<td>.gnu.version_r</td>
<td>SHT_GNU_verneed</td>
<td>SHF_ALLOCS</td>
</tr>
<tr>
<td>.got.plt</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOCS+SHF_WRITE</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.
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 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>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<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>
</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_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>

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_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 boundary.</td>
</tr>
<tr>
<td>Instruction</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<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 representing a register number and a signed LEB128 factored offset. This instruction is identical to DW_CFA_offset_extended except that the second operand is signed.</td>
</tr>
<tr>
<td>DW_CFA_def_cfa_sf</td>
<td>0x12</td>
<td>The DW_CFA_def_cfa_sf instruction takes two operands: an unsigned</td>
</tr>
<tr>
<td>Instruction</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DW_CFA_def_cfa_offset_sf</td>
<td>0x13</td>
<td>The DW_CFA_def_cfa_offset_sf instruction takes a signed LEB128 operand representing a factored offset. This instruction is identical to DW_CFA_def_cfa except that the second operand is signed and factored.</td>
</tr>
<tr>
<td>DW_CFA_GNU_args_size</td>
<td>0x2e</td>
<td>The DW_CFA_GNU_args_size 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>
</tbody>
</table>
| DW_CFA_GNU_negative_offset_extended | 0x2f  | The DW_CFA_def_cfa_sf instruction takes two operands: an unsigned LEB128 value representing a register number and an unsigned LEB128 which represents the magnitude of the offset. This instruction is identical to DW_CFA_offset_extended_sf except that the operand is subtracted to produce the offset. This instruction is obsoleted by DW_CFA_offset_extended.
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>
<th>Frame Description Entry Record(s)</th>
</tr>
</thead>
</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>
<tr>
<td>Code Alignment Factor</td>
<td>Required</td>
</tr>
<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:
11 Object Format

ISO/IEC 23360 Part 1:2008(E)

'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>
<thead>
<tr>
<th>Encoding</th>
<th>Field</th>
</tr>
</thead>
</table>
11 Object Format

<table>
<thead>
<tr>
<th>unsigned byte</th>
<th>version</th>
</tr>
</thead>
<tbody>
<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

© 2008 Linux Foundation
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_Verdaux 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_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.
11 Object Format

ISO/IEC 23360 Part 1:2008(E)

### 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.

```c
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.

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

**Figure 11-3 Version Needed Entries**

#### 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>
<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;
```
12 Dynamic Linking

ISO/IEC 23360 Part 1:2008(E)

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_RELA
  Address of Rela relocations

DT_RELAENT
  Size of one Rela reloc

DT_RELASZ
  Total size of Rela relocations

DT_RELENT
  Size of one Rel reloc

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_SYMBOLIC
  Start symbol search here

DT_SYMENT
  Size of one symbol table entry

DT_SYMTAB
  Address of symbol table

DT_TEXTREL
  Reloc might modify .text
12 Dynamic Linking

12.3.2.2 Additional Dynamic Entries

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

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_SYMIMENT
   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 as 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>Table 13-1 libc Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library:</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
[RPC & XDR] RFC 1831/1832 RPC & XDR
[SUSv2] SUSv2
[SUSv4] POSIX 1003.1 2008
[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>clnt_serror [SVID.4]</td>
<td>clnt_raw_create</td>
<td>clnttcp_create</td>
<td>clntudp_bufcreate</td>
</tr>
<tr>
<td>clntudp_create [RPC &amp; XDR]</td>
<td>key_decryptsession</td>
<td>pmap_getport</td>
<td>pmap_set [LSB]</td>
</tr>
<tr>
<td>pmap_unset [LSB]</td>
<td>svc_getreqset</td>
<td>svc_register</td>
<td>svc_run [LSB]</td>
</tr>
<tr>
<td>svc_sendreply [LSB]</td>
<td>svcerr_auth</td>
<td>svcerr_decode</td>
<td>svcerr_nopro c [SVID.3]</td>
</tr>
<tr>
<td>svcerr_nopro g [SVID.3]</td>
<td>svcerr_progres</td>
<td>svcerr_systemerr</td>
<td>svcerr_weakauth [SVID.3]</td>
</tr>
<tr>
<td>svcfd_create [RPC &amp; XDR]</td>
<td>svcraw_create</td>
<td>svctcp_create</td>
<td>svcudp_create [LSB]</td>
</tr>
<tr>
<td>xdr_accepted_reply [SVID.3]</td>
<td>xdr_array</td>
<td>xdr_bool</td>
<td>xdr_bytes [SVID.3]</td>
</tr>
<tr>
<td>xdr_callhdr [SVID.3]</td>
<td>xdr_callmsg</td>
<td>xdr_char</td>
<td>xdr_double [SVID.3]</td>
</tr>
<tr>
<td>xdr_enum [SVID.3]</td>
<td>xdr_float</td>
<td>xdr_free</td>
<td>xdr_int [SVID.3]</td>
</tr>
<tr>
<td>xdr_long [SVID.3]</td>
<td>xdr_opaque</td>
<td>xdr_opaque_auth</td>
<td>xdr_pointer [SVID.3]</td>
</tr>
<tr>
<td>xdr_reference [SVID.3]</td>
<td>xdr_rejected_reply</td>
<td>xdr_replaymsg</td>
<td>xdr_short [SVID.3]</td>
</tr>
<tr>
<td>xdr_string</td>
<td>xdr_u_char</td>
<td>xdr_u_int [LSB]</td>
<td>xdr_u_long</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for RPC specified in Table 13-3, 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 13-3 libc - RPC Deprecated Function Interfaces

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

### 13.3.2 Epoll

#### 13.3.2.1 Interfaces for Epoll

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

### Table 13-4 libc - Epoll Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>epoll_create(GLIBC_2.3.2)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>epoll_ctl(GLIBC_2.3.2)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>epoll_wait(GLIBC_2.3.2)</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

### 13.3.3 System Calls

#### 13.3.3.1 Interfaces for System Calls

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

### Table 13-5 libc - System Calls Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>__chk_fail(GLIBC C_2.3.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__fxstat [LSB]</td>
<td></td>
</tr>
<tr>
<td>__fxstatat(GLIBC C_2.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__getgroups_chk(GLIBC_2.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__getpgid [LSB]</td>
<td></td>
</tr>
<tr>
<td>__lxstat [LSB]</td>
<td></td>
</tr>
<tr>
<td>__read_chk(GLIBC_2.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__readlink_chk(GLIBC_2.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__stack_chk_fail(GLIBC_2.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__xmknod [LSB]</td>
<td></td>
</tr>
<tr>
<td>__xmknodat(GLIBC_2.4)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__xstat [LSB]</td>
<td></td>
</tr>
<tr>
<td>access [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>acct [LSB]</td>
<td></td>
</tr>
<tr>
<td>alarm [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>brk [SUSv2]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>SUSv3, SUSv4, LSB, Glibc 2.3.4, GLIBC 2.4</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>chdir</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>clock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>dup</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>execl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>execv</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>exit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fchdir</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fchown</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fchownat</td>
<td>[GLIBC 2.4]</td>
</tr>
<tr>
<td>fchroot</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fchrootat</td>
<td>[GLIBC 2.4]</td>
</tr>
<tr>
<td>fdatasync</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fdopen</td>
<td>[GLIBC 2.4]</td>
</tr>
<tr>
<td>fexecve</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fexecvp</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fork</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fstatfs</td>
<td>[LSB]</td>
</tr>
<tr>
<td>fstatvfs</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ftruncate</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getcontext</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getdtablesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>getegid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>geteuid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getgid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getgroups</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getgroups</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getitimer</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getloadavg</td>
<td>[LSB]</td>
</tr>
<tr>
<td>getpagesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>getpgid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getpriority</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getpriority</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getrlimit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getrusage</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getsid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getuid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getwd</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>initgroups</td>
<td>[LSB]</td>
</tr>
<tr>
<td>ioctl</td>
<td>[LSB]</td>
</tr>
<tr>
<td>kill</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lckf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lseek</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>mkdir</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>mkfifo</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>mknod</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>mprotect</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>munlock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>munlockall</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>munmap</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>munmap</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nice</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>open</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>openat</td>
<td>[GLIBC 2.4]</td>
</tr>
<tr>
<td>pathconf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pause</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pipe</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>poll</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pselect</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>read</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>readdir</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>readdir_r</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>readdir</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>readlink</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>readlinkat</td>
<td>[GLIBC 2.4]</td>
</tr>
<tr>
<td>readv</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>rename</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>remap</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>rmdir</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sbk</td>
<td>[SUSv2]</td>
</tr>
<tr>
<td>sched_get_priority_max</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_get_priority_min</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_getparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_getscheduler</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_set_get_priority_max</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_set_get_priority_min</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_setparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_setscheduler</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_yield</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Technology</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>sched_rr_get_interval</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_setaffinity</td>
<td>(GLIBC_2.3.4)</td>
</tr>
<tr>
<td>sched_setparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sched_setscheduler</td>
<td>[LSB]</td>
</tr>
<tr>
<td>sched_yield</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>select</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setcontext</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setegid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setgid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setitimer</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setpgid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setpriority</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setrlimit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setrlimit64</td>
<td>[LFS]</td>
</tr>
<tr>
<td>setsid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setuid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sleep</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>statfs</td>
<td>[LSB]</td>
</tr>
<tr>
<td>statvfs</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sync</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sysconf</td>
<td>[LSB]</td>
</tr>
<tr>
<td>time</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>times</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>truncate</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ulimit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>umask</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>uname</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>unslink</td>
<td>[LSB]</td>
</tr>
<tr>
<td>unlinkat</td>
<td>(GLIBC_2.4)</td>
</tr>
<tr>
<td>utime</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>utimes</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>vfork</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>wait</td>
<td>[LSB]</td>
</tr>
<tr>
<td>waitid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>waitpid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>write</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>writev</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic deprecated functions for System Calls specified in Table 13-6, 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 13-6 libc - System Calls Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>fstatfs</td>
<td>[LSB]</td>
</tr>
<tr>
<td>getdtablesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>getpagesize</td>
<td>[LSB]</td>
</tr>
<tr>
<td>getwd</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>statfs</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

### 13.3.4 Standard I/O

#### 13.3.4.1 Interfaces for Standard I/O

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

### Table 13-7 libc - Standard I/O Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IO_feof</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_IO_getc</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_IO_putc</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_IO_puts</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__fgets_chk</td>
<td>(GLIBC_2.4)</td>
</tr>
<tr>
<td>__fgets_unlocked_chk</td>
<td>(GLIBC_2.4)</td>
</tr>
<tr>
<td>__fprintf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__printf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__snprintf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__sprintf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__vfprintf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__vprintf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__vsprintf_chk</td>
<td>[LSB]</td>
</tr>
<tr>
<td>asprintf</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
An LSB conforming implementation shall provide the generic deprecated functions for Standard I/O 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 - Standard I/O Deprecated Function Interfaces

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

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

Table 13-9 libc - Standard I/O Data Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>stderr</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>stdin</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>stdout</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

### 13.3.5 Signal Handling

#### 13.3.5.1 Interfaces for Signal Handling

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

Table 13-10 libc - Signal Handling Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>__libc_current_sigl</td>
</tr>
<tr>
<td>__libc_current_sigh</td>
</tr>
<tr>
<td>__sigsetjmp</td>
</tr>
<tr>
<td>__sysv_signal</td>
</tr>
<tr>
<td>__xpg_sigpause</td>
</tr>
<tr>
<td>__sigsetjmp</td>
</tr>
<tr>
<td>__sigaction</td>
</tr>
<tr>
<td>__sigaddset</td>
</tr>
<tr>
<td>__sigaltstack</td>
</tr>
<tr>
<td>__sigdelset</td>
</tr>
<tr>
<td>__sigemptyset</td>
</tr>
<tr>
<td>__siginterrupt</td>
</tr>
<tr>
<td>__siglongjmp</td>
</tr>
<tr>
<td>__sigpending</td>
</tr>
<tr>
<td>__sigprocmask</td>
</tr>
<tr>
<td>__sigqueue</td>
</tr>
<tr>
<td>__sigreturn</td>
</tr>
<tr>
<td>__sigset</td>
</tr>
<tr>
<td>__sigsuspend</td>
</tr>
<tr>
<td>__sigtimedwait</td>
</tr>
<tr>
<td>__sigwait</td>
</tr>
<tr>
<td>__sigwaitinfo</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic deprecated functions for Signal Handling specified in Table 13-11, 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 13-11 libc - Signal Handling Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigpause</td>
</tr>
<tr>
<td>[LSB]</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Signal Handling specified in Table 13-12, with the full mandatory functionality.
as described in the referenced underlying specification.

### Table 13-12 libc - Signal Handling Data Interfaces

<table>
<thead>
<tr>
<th>_sys_siglist</th>
<th>[LSB]</th>
</tr>
</thead>
</table>

### 13.3.6 Localization Functions

#### 13.3.6.1 Interfaces for Localization Functions

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

### Table 13-13 libc - Localization Functions Function Interfaces

<table>
<thead>
<tr>
<th>bind_textdomain_codeset</th>
<th>bindtextdomain</th>
<th>catclose</th>
<th>catgets</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>catopen</td>
<td>dcgettext</td>
<td>dcngettext</td>
<td>dgettext</td>
</tr>
<tr>
<td>[USv3]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
</tr>
<tr>
<td>dngettext</td>
<td>duplocale(GLIBC_2.3)</td>
<td>freelocale(GLIBC_2.3)</td>
<td>gettext</td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
</tr>
<tr>
<td>iconv</td>
<td>iconv_close</td>
<td>iconv_open</td>
<td>localeconv</td>
</tr>
<tr>
<td>[USv3]</td>
<td>[USv3]</td>
<td>[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>newlocale(GLIBC_2.3)</td>
<td>ngettext</td>
<td>nl_langinfo</td>
<td>setlocale</td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>textdomain</td>
<td>uselocale(GLIBC_2.3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td></td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Localization Functions specified in Table 13-14, with the full mandatory functionality as described in the referenced underlying specification.

### Table 13-14 libc - Localization Functions Data Interfaces

<table>
<thead>
<tr>
<th>_nl_msg_cat_cnt</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LSB]</td>
<td></td>
</tr>
</tbody>
</table>

### 13.3.7 Posix Spawn Option

#### 13.3.7.1 Interfaces for Posix Spawn Option

An LSB conforming implementation shall provide the generic functions for Posix Spawn Option specified in Table 13-15, with the full mandatory functionality as described in the referenced underlying specification.

### Table 13-15 libc - Posix Spawn Option Function Interfaces

<table>
<thead>
<tr>
<th>posix_spawn</th>
<th>posix_spawn_file_actions_addclo</th>
<th>posix_spawn_file_actions_addup2</th>
<th>posix_spawn_file_actions_addopen</th>
</tr>
</thead>
<tbody>
<tr>
<td>[USv3]</td>
<td>[USv3]</td>
<td>[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>posix_spawn_file_actions_destroys</td>
<td>posix_spawn_file_actions_init</td>
<td>posix_spawnattr_destroy</td>
<td>posix_spawnattr_getflags</td>
</tr>
<tr>
<td>[USv3]</td>
<td>[USv3]</td>
<td>[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>posix_spawnattr</td>
<td>posix_spawnattr</td>
<td>posix_spawnattr</td>
<td>posix_spawnattr</td>
</tr>
</tbody>
</table>
13.3.8 Posix Advisory Option

13.3.8.1 Interfaces for Posix Advisory Option

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

Table 13-16 libc - Posix Advisory Option Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>posix_fadvise</td>
<td></td>
</tr>
<tr>
<td>posix_fallocate</td>
<td></td>
</tr>
<tr>
<td>posix_madvise</td>
<td></td>
</tr>
<tr>
<td>posix_memalign</td>
<td></td>
</tr>
</tbody>
</table>

13.3.9 Socket Interface

13.3.9.1 Interfaces for Socket Interface

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

Table 13-17 libc - Socket Interface Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>bind</td>
<td></td>
</tr>
<tr>
<td>bindresvport</td>
<td></td>
</tr>
<tr>
<td>connect</td>
<td></td>
</tr>
<tr>
<td>gethostid</td>
<td></td>
</tr>
<tr>
<td>gethostname</td>
<td></td>
</tr>
<tr>
<td>getpeername</td>
<td></td>
</tr>
<tr>
<td>getsockname</td>
<td></td>
</tr>
<tr>
<td>getsockopt</td>
<td></td>
</tr>
<tr>
<td>if_freenameindex</td>
<td></td>
</tr>
<tr>
<td>if_indextoname</td>
<td></td>
</tr>
<tr>
<td>if_nameindex</td>
<td></td>
</tr>
<tr>
<td>if_nametoindex</td>
<td></td>
</tr>
<tr>
<td>listen</td>
<td></td>
</tr>
<tr>
<td>recv</td>
<td></td>
</tr>
<tr>
<td>recvfrom</td>
<td></td>
</tr>
<tr>
<td>recvmsg</td>
<td></td>
</tr>
<tr>
<td>send</td>
<td></td>
</tr>
<tr>
<td>sendmsg</td>
<td></td>
</tr>
<tr>
<td>sendto</td>
<td></td>
</tr>
<tr>
<td>shutdown</td>
<td></td>
</tr>
<tr>
<td>socket</td>
<td></td>
</tr>
<tr>
<td>socketpair</td>
<td></td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic data interfaces for Socket Interface specified in Table 13-18, with the full mandatory functionality as described in the referenced underlying specification.

### Table 13-18 libc - Socket Interface Data Interfaces

<table>
<thead>
<tr>
<th>in6 addr _any</th>
<th>in6 addr _loopbac k</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SUSv3]</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

### 13.3.10 Wide Characters

#### 13.3.10.1 Interfaces for Wide Characters

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

### Table 13-19 libc - Wide Characters Function Interfaces

<table>
<thead>
<tr>
<th>__fgetws _chk(GLIBC_2.4) [LSB]</th>
<th>__fwprintf _chk(GLIBC_2.4) [LSB]</th>
<th>__mbsnrtowcs _chk(GLIBC_2.4) [LSB]</th>
<th>__mbstowcs _chk(GLIBC_2.4) [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>__mbsnrtombs _chk(GLIBC_2.4) [LSB]</td>
<td>__wcsnrtowcs _chk(GLIBC_2.4) [LSB]</td>
<td>__wcstod _internal [LSB]</td>
<td>__wcstol _internal [LSB]</td>
</tr>
<tr>
<td>__wcstold _internal [LSB]</td>
<td>__wctomb _chk(GLIBC_2.4) [LSB]</td>
<td>__wctomb _chk(GLIBC_2.4) [LSB]</td>
<td>__wctomb _chk(GLIBC_2.4) [LSB]</td>
</tr>
<tr>
<td>__wcstold _internal [LSB]</td>
<td>__wmemmove _chk(GLIBC_2.4) [LSB]</td>
<td>__wmemmove _chk(GLIBC_2.4) [LSB]</td>
<td>__wmemmove _chk(GLIBC_2.4) [LSB]</td>
</tr>
<tr>
<td>__wmemset _chk(GLIBC_2.4) [LSB]</td>
<td>__wprintf _chk(GLIBC_2.4) [LSB]</td>
<td>__fwscanf [SUSv3]</td>
<td>__fgetwc [SUSv3]</td>
</tr>
</tbody>
</table>
13.3.11 String Functions

13.3.11.1 Interfaces for String Functions

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__memcpy_chk</td>
<td>GLIBC_2.3.4</td>
<td>__memmove_chk</td>
</tr>
<tr>
<td>__memmove_chk</td>
<td>GLIBC_2.3.4</td>
<td>__memcpy</td>
</tr>
<tr>
<td>__memset_chk</td>
<td>GLIBC_2.3.4</td>
<td>__memmove_chk</td>
</tr>
<tr>
<td>__memmove_chk</td>
<td>GLIBC_2.3.4</td>
<td>__memcpy</td>
</tr>
<tr>
<td>__memcpy</td>
<td>GLIBC_2.3.4</td>
<td>__memmove_chk</td>
</tr>
<tr>
<td>__memmove</td>
<td>GLIBC_2.3.4</td>
<td>__memcpy</td>
</tr>
<tr>
<td>__memset</td>
<td>GLIBC_2.3.4</td>
<td>__memmove_chk</td>
</tr>
<tr>
<td>__memmove</td>
<td>GLIBC_2.3.4</td>
<td>__memcpy</td>
</tr>
<tr>
<td>__memcpy</td>
<td>GLIBC_2.3.4</td>
<td>__memmove_chk</td>
</tr>
<tr>
<td>__memmove</td>
<td>GLIBC_2.3.4</td>
<td>__memcpy</td>
</tr>
</tbody>
</table>

Table 13-20 libc - String Functions Function Interfaces

© 2008 Linux Foundation
An LSB conforming implementation shall provide the generic deprecated functions for String Functions specified in Table 13-21, 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>_stpcpy_chk(GLIBC_2.4) [LSB]</th>
<th>_strcat_chk(GLIBC_2.3.4) [LSB]</th>
<th>_strncpy_chk(GLIBC_2.3.4) [LSB]</th>
<th>_strdup [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>_strncat_chk(GLIBC_2.3.4) [LSB]</td>
<td>_strcpy_chk(GLIBC_2.3.4) [LSB]</td>
<td>_strtof_internal [LSB]</td>
<td>_strtoc内部 [LSB]</td>
</tr>
<tr>
<td>_strtok_r [LSB]</td>
<td>_strtol_internal [LSB]</td>
<td>_strtold_internal [LSB]</td>
<td>_strtoll_internal [LSB]</td>
</tr>
<tr>
<td>_strtoul_internal [LSB]</td>
<td>_strtoq [LSB]</td>
<td>_xpg_strerror_r (GLIBC_2.3.4) [LSB]</td>
<td>bcmp [SUSv3]</td>
</tr>
<tr>
<td>swab [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 13-21 libc - String Functions Deprecated Function Interfaces

| strerror_r [LSB] |

### 13.3.12 IPC Functions

#### 13.3.12.1 Interfaces for IPC Functions

An LSB conforming implementation shall provide the generic functions for IPC Functions specified in Table 13-22, with the full mandatory functionality as de-
scribed in the referenced underlying specification.

Table 13-22 libc - IPC Functions Function Interfaces


13.3.13 Regular Expressions

13.3.13.1 Interfaces for Regular Expressions

An LSB conforming implementation shall provide the generic functions for Regular Expressions specified in Table 13-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-23 libc - Regular Expressions Function Interfaces


13.3.14 Character Type Functions

13.3.14.1 Interfaces for Character Type Functions

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

Table 13-24 libc - Character Type Functions Function Interfaces

| __ctype_b_loc(GLIBC_2.3) [LSB] | __ctype_get_mb_cur_max [LSB] | __ctype_tolower_loc(GLIBC_2.3) [LSB] | __ctype_toupper_loc(GLIBC_2.3) [LSB] |
| tolower [SUSv3] | toupper [SUSv3] |

13.3.15 Time Manipulation

13.3.15.1 Interfaces for Time Manipulation

An LSB conforming implementation shall provide the generic functions for Time Manipulation specified in Table 13-25, with the full mandatory functional-
ity as described in the referenced underlying specification.

**Table 13-25 libc - Time Manipulation Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjtime</td>
<td>[LSB]</td>
<td>asctime</td>
<td>[SUSv3]</td>
<td>asctime</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>time_r</td>
<td>[SUSv3]</td>
<td>difftime</td>
<td>[SUSv3]</td>
<td>gmtime</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
<td>gmtime_r</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>localtime</td>
<td>[SUSv3]</td>
<td>localtime_r</td>
<td>[SUSv3]</td>
<td>mktime</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>tzset</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></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 13-26**, with the full mandatory functionality as described in the referenced underlying specification.

**Table 13-26 libc - Time Manipulation Data Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>__daylight</td>
<td>[LSB]</td>
<td>__timezone</td>
<td>[LSB]</td>
<td>__tzname</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

### 13.3.16 Terminal Interface Functions

#### 13.3.16.1 Interfaces for Terminal Interface Functions

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

**Table 13-27 libc - Terminal Interface Functions Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
<td>cfsetispeed</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>cfsetspeed</td>
<td>[SUSv3]</td>
<td>cfsetspeed</td>
<td>[LSB]</td>
<td>tcdrain</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
<td>tcflow</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>tcflush</td>
<td>[SUSv3]</td>
<td>tcgetattr</td>
<td>[SUSv3]</td>
<td>tcgetpgrp</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
<td>tcgetsid</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>tcsendbreak</td>
<td>[SUSv3]</td>
<td>tcsetattr</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 13.3.17 System Database Interface

#### 13.3.17.1 Interfaces for System Database Interface

An LSB conforming implementation shall provide the generic functions for System Database Interface specified in **Table 13-28**, with the full mandatory functionality as described in the referenced underlying specification.

**Table 13-28 libc - System Database Interface Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
<th>Function</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>endgrent</td>
<td>[SUSv3]</td>
<td>endprotoent</td>
<td>[SUSv3]</td>
<td>endpwent</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
<td>endservent</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>endutent</td>
<td>[LSB]</td>
<td>endutxent</td>
<td>[SUSv3]</td>
<td>getgrent</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>getgrent_r</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for System Database Interface 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 13-29 libc - System Database Interface Deprecated Function Interfaces

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>gethostbyname2_r [LSB]</td>
<td>gethostbyname2_r [LSB]</td>
<td>gethostbyname_r [SUSv3]</td>
<td>gethostbyname_r [SUSv3]</td>
</tr>
</tbody>
</table>

### 13.3.18 Language Support

#### 13.3.18.1 Interfaces for Language Support

An LSB conforming implementation shall provide the generic functions for Language Support specified in Table 13-30, with the full mandatory functionality as described in the referenced underlying specification.

### Table 13-30 libc - Language Support Function Interfaces

| __libc_start_thenReturn [LSB] | __register_atfork (GLIBC_2.3.2) [LSB] |
13.3.19 Large File Support

13.3.19.1 Interfaces for Large File Support

An LSB conforming implementation shall provide the generic functions for Large File Support specified in Table 13-31, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-31 libc - Large File Support Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>GLIBC_2.4</th>
<th>LSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>__fxstat64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__fxstatat64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xstat64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__lxstat64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xstat64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>creat64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetpos64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fopen64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>freopen64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fseeko64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fsetpos64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fstatfs64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fstatvfs64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ftello64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ftw64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ftw64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ftw64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getrlimit64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lockf64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mkstemp64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mmap64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>openat64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>posix_fadvise64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>posix_fallocate64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readdir64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>readdir64_r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpfile64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncate64</td>
<td>LFS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fstatfs64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>statfs64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>statvfs64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpfile64</td>
<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 13-32, 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 13-32 libc - Large File Support Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>fstatfs64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>statfs64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.3.20 Inotify

13.3.20.1 Interfaces for Inotify

An LSB conforming implementation shall provide the generic functions for Inotify specified in Table 13-33, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-33 libc - Inotify Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>LSB</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>inotify_add_watch(GLIBC_2.4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inotify_init(GLIBC_2.4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inotify_rm_watch(GLIBC_2.4)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.3.21 Standard Library

13.3.21.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the generic functions for Standard Library specified in Table 13-34, with the full mandatory functionality as described in the referenced underlying specification.
<table>
<thead>
<tr>
<th>Function</th>
<th>Implementation</th>
<th>Function</th>
<th>Implementation</th>
<th>Function</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Exit [SUSv3]</td>
<td>_assert_fail [LSB]</td>
<td>_confstr_chk(GLIBC_2.4) [LSB]</td>
<td>_cxa_atexit [LSB]</td>
<td>__cxa_finalize [LSB]</td>
<td></td>
</tr>
<tr>
<td>__cxa_finalize [LSB]</td>
<td>_errno_location [LSB]</td>
<td>__fpending [LSB]</td>
<td>__getcwd_chk(GLIBC_2.4) [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__getlogin_r_chk(GLIBC_2.4) [LSB]</td>
<td>getpagesize [LSB]</td>
<td>__isinf [LSB]</td>
<td>__isinf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__isinf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__isinf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__isninf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__isninf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__isninf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__isninf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td>__isnaf [LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__pcre64_chk(GLIBC_2.4) [LSB]</td>
<td>__realpath_chk(GLIBC_2.4) [LSB]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__syslog_chk(GLIBC_2.4) [LSB]</td>
<td>__ttname_r_chk(GLIBC_2.4) [LSB]</td>
<td>__vsyslog_chk(GLIBC_2.4) [LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>glob64 [LSB]</td>
<td>globfree [SUSv3]</td>
<td>globfree64 [LSB]</td>
<td>granpt [SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inet_ntoa</td>
<td>inet_ntop</td>
<td>inet_pton</td>
<td>initstate [SUSv3]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13 Base Libraries

An LSB conforming implementation shall provide the generic deprecated functions for Standard Library 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.
ISO/IEC 23360 Part 1:2008(E)

Table 13-35 libc - Standard Library Deprecated Function Interfaces


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

Table 13-36 libc - Standard Library Data Interfaces

| __environ [LSB] | __environ [LSB] | _sys_errlist [LSB] | environ [SUSv3] |
| optopt [SUSv3] |

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 *__cp);
extern int inet_aton(const char *__cp, struct in_addr *__inp);
extern char *inet_ntoa(struct in_addr __in);
extern const char *inet_ntop(int __af, const void *__cp, char *

13.4.2 assert.h

#define NDEBUG
#define assert(expr) ((void)0)
#else
#define assert(expr) ((void) ((expr) ? 0 : (__assert_fail (#expr, __FILE__, __LINE__, __PRETTY_FUNCTION__), 0)))

© 2008 Linux Foundation
#endif
extern void __assert_fail(const char *__assertion, const char *__file, unsigned int __line, const char *__function);

13.4.3 cpio.h

#define C_IXOTH 000001
#define C_IWOTH 000002
#define C_IROTH 000004
#define C_IXGRP 000010
#define C_IWGRP 000020
#define C_IRGRP 000040
#define C_IXUSR 000100
#define C_IWUSR 000200
#define C_IRUSR 000400
#define C_ISVTX 000800
#define C_ISGID 001000
#define C_ISUID 002000
#define C_ISVTX        010000
#define C_ISREG 0100000
#define C_ISCTG 0110000
#define C_ISLNK 0120000
#define C_ISSOCK        0140000
#define C_ISCHR 020000
#define C_ISDIR 040000
#define C_ISBLK 060000
#define MAGIC   "070707"

13.4.4 ctype.h

extern const unsigned short **__ctype_b_loc(void);
extern const int32_t **__ctype_tolower_loc(void);
extern const int32_t **__ctype_toupper_loc(void);
extern int _tolower(int);
extern int _toupper(int);
extern int isalnum(int);
extern int isalpha(int);
extern int isascii(int __c);
extern int isblank(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 __c);
extern int tolower(int __c);
extern int toupper(int __c);

13.4.5 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 alphasort(const struct dirent **__e1,
    const struct dirent **__e2);
extern int alphasort64(const struct dirent64 **__e1,
    const struct dirent64 **__e2);
extern int closedir(DIR * __dirp);
extern int dirfd(DIR * __dirp);
extern DIR *fdopendir(int __fd);
extern DIR *opendir(const char *__name);
extern struct dirent *readdir(DIR * __dirp);
extern struct dirent64 *readdir64(DIR * __dirp);
extern int readdir64_r(DIR * __dirp, struct dirent64 *__entry,
    struct dirent64 **__result);
extern int readdir_r(DIR * __dirp, struct dirent *__entry,
    struct dirent **__result);
extern void rewinddir(DIR * __dirp);
extern int scandir(const char *__dir, struct dirent***__namelist,
    int (*__selector) (const struct dirent *),
    int (*__cmp) (const struct dirent *,
        const struct dirent *));
extern int scandir64(const char *__dir, struct dirent64***__namelist,
    int (*__selector) (const struct dirent64 *),
    int (*__cmp) (const struct dirent64 *,
        const struct dirent64 *));
extern void seekdir(DIR * __dirp, long int __pos);
extern long int telldir(DIR * __dirp);

13.4.6 endian.h

#define __LITTLE_ENDIAN 1234
#define __BIG_ENDIAN 4321
#define BIG_ENDIAN __BIG_ENDIAN
#define BYTE_ORDER __BYTE_ORDER
#define LITTLE_ENDIAN __LITTLE_ENDIAN

13.4.7 err.h

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

13.4.8 errno.h

#define errno (*__errno_location())
#define EPERM 1 /* Operation not permitted */
#define ECHILD 10 /* No child processes */
#define ENETDOWN 100 /* Network is down */
#define ENETUNREACH 101 /* Network is unreachable */
#define ENETRESET 102 /* Network dropped connection because of reset */
#define ECONNABORTED 103 /* Software caused connection abort */
#define ECONNRESET 104 /* Connection reset by peer */
#define ENOBUFS 105 /* No buffer space available */
#define EISCONN 106 /* Transport endpoint is already connected */
#define ENOTCONN 107 /* Transport endpoint is not connected */
#define ESHUTDOWN 108 /* Cannot send after transport endpoint shutdown */
#define ETIMEOUT 109 /* Too many references: cannot splice */
#define EAGAIN 11 /* Try again */
#define ETIMEDOUT 110 /* Connection timed out */
#define ECONNREFUSED 111 /* Connection refused */
#define EHOSTDOWN 112 /* Host is down */
#define EHOSTUNREACH 113 /* No route to host */
#define EALREADY 114 /* Operation already in progress */
#define EINPROGRESS 115 /* Operation now in progress */
#define ENOTNAM 116 /* Not a XENIX named type file */
#define EREMOTE 117 /* Remote file */
#define EUCLEAN 118 /* Structure needs cleaning */
#define ENOTNAM 119 /* Not a XENIX named type file */
#define ENOMEM 12 /* Out of memory */
#define EISNAM 120 /* Is a named type file */
#define EREMOTE 121 /* Remote I/O error */
#define EDQUOT 122 /* Quota exceeded */
#define ENOREM 123 /* No medium found */
#define ENOSTAT 124 /* Wrong medium type */
#define ECANCELED 125 /* Operation Canceled */
#define ENOSPC 126 /* No such device */
#define ECONFLICT 127 /* No such file or directory */
#define EDIR 128 /* Is a directory */
#define EINVAL 129 /* Invalid argument */
#define ENFILE 130 /* File table overflow */
#define EMT 131 /* Too many open files */
#define ENOTTY 132 /* Not a typewriter */
#define ETXTBSY 133 /* Text file busy */
#define EFBIG 134 /* File too large */
#define ENOSPC 135 /* No space left on device */
#define EPIPE 136 /* Broken pipe */
#define EDOM 137 /* Math argument out of domain of func */
#define ERANGE 138 /* Math result not representable */
#define EDEADLK 139 /* Resource deadlock would occur */
#define ENAMETOOLONG 140 /* File name too long */
ISO/IEC 23360 Part 1:2008(E)

#define ENOLCK 37              /* No record locks available */
#define ENOSYS 38               /* Function not implemented */
#define ENOTEMPTY 39            /* Directory not empty */
#define EINTR 4                  /* Interrupted system call */
#define ELOOP 40                 /* Too many symbolic links encountered */
#define ENOMSG 42                /* No message of desired type */
#define EIDRM 43                 /* Identifier removed */
#define ECHRNG 44                /* Channel number out of range */
#define EL2NSYNC 45              /* Level 2 not synchronized */
#define EL3HLT 46                /* Level 3 halted */
#define EL3RST 47                /* Level 3 reset */
#define ELNRMNG 48               /* Link number out of range */
#define EUNATCH 49               /* Protocol driver not attached */
#define EIO 5                    /* I/O error */
#define ENOANO 55                /* No anode */
#define EBADRQC 56               /* Invalid request code */
#define EBADSLT 57               /* Invalid slot */
#define EBFONT 59                /* Bad font file format */
#define ENXIO 6                  /* No such device or address */
#define ENOSTR 60                /* Device not a stream */
#define ENODATA 61               /* No data available */
#define ETIME 62                 /* Timer expired */
#define ENOSR 63                 /* Out of streams resources */
#define ENONET 64                /* Machine is not on the network */
#define ENOPKG 65                /* Package not installed */
#define EREMOTE 66               /* Object is remote */
#define ENOLINK 67               /* Link has been severed */
#define EADV 68                  /* Advertise error */
#define ESRMNT 69                /* Srmount error */
#define E2BIG 7                  /* Argument list too long */
#define ECOMM 70                 /* Communication error on send */
#define EPROTO 71                /* Protocol error */
#define EMULTIHOP 72             /* Multihop attempted */
#define EDESTADDRREQ 89          /* Destination address required */
#define EBADF 9                  /* Bad file number */
#define EMSGSIZE 90              /* Message too long */

© 2008 Linux Foundation
#define EPROTOTYPE  91      /* Protocol wrong type for socket */
#define ENOPROTOOPT 92      /* Protocol not available */
#define EPROTONOSUPPORT 93      /* Protocol not supported */
#define ESOCKTNOSUPPORT 94      /* Socket type not supported */
#define EOPNOTSUPP  95      /* Operation not supported on transport endpoint */
#define EPFNOSUPPORT 96      /* Protocol family not supported */
#define EAFNOSUPPORT 97      /* Address family not supported by protocol */
#define EADDRINUSE  98      /* Address already in use */
#define EADDRNOTAVAIL 99      /* Cannot assign requested address */
#define EWOULDBLOCK EAGAIN  /* Operation would block */
#define ENOTSUP EOPNOTSUPP

extern int *__errno_location(void);

13.4.9 fcntl.h

#define POSIX_FADV_NORMAL 0
#define O_RDONLY        00
#define O_ACCMODE       0003
#define O_WRONLY        01
#define O_CREAT 0100
#define O_TRUNC 01000
#define O_DSYNC 010000
#define O_RSYNC 010000
#define O_SYNC  010000
#define O_RDWR  02
#define O_EXCL  0200
#define O_APPEND        02000
#define O_ASYNC 020000
#define O_NDELAY        04000
#define O_NONBLOCK      04000
#define FD_CLOEXEC      1
#define POSIX_FADV_RANDOM 1
#define POSIX_FADV_SEQUENTIAL 2
#define POSIX_FADV_WILLNEED 3

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

#define AT_FDCWD -100
#define AT_SYMLINK_NOFOLLOW 0x100
#define AT_EACCESS 0x200
#define AT_REMOVEDIR 0x200
#define AT_SYMLINK_FOLLOW 0x400

#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 creat(const char *__file, mode_t __mode);
extern int creat64(const char *__file, mode_t __mode);
extern int fcntl(int __fd, int __cmd, ...);
extern int open(const char *__file, int __oflag, ...);
extern int open64(const char *__file, int __oflag, ...);
extern int openat(int __fd, const char *__file, int __oflag, ...);
extern int openat64(int __fd, const char *__file, int __oflag, ...);
extern int posix_fadvise(int __fd, off_t __offset, off_t __len, int __advise);
extern int posix_fadvise64(int __fd, off64_t __offset, off64_t __len, int __advise);
extern int posix_fallocate(int __fd, off_t __offset, off_t __len);
extern int posix_fallocate64(int __fd, off64_t __offset, off64_t __len);

13.4.10 fmtmsg.h

#define MM_HARD 1 /* Source of the condition is hardware. */
#define MM_NRECOV 128 /* Non-recoverable error. */
#define MM_UTIL 16 /* Condition detected by utility. */
#define MM_SOFT 2 /* Source of the condition is software. */
#define MM_PRINT 256 /* Display message in standard error. */
#define MM_OPSYS 32 /* Condition detected by operating system. */
#define MM_FIRM 4 /* Source of the condition is firmware. */
#define MM_CONSOLE 512 /* Display message on system console. */
#define MM_RECOVER 64 /* Recoverable error. */
#define MM_APPL 8 /* Condition detected by application. */
#define MM_NOSEV 0 /* No severity level provided for the message. */
#define MM_HALT 1 /* Error causing application to halt. */
#define MM_ERROR 2 /* Application has encountered a non-fatal fault. */
#define MM_WARNING 3 /* Application has detected unusual non-error condition. */
#define MM_INFO 4 /* Informative message. */
#define MM_NULLACT ((char *) 0)
#define MM_NULLLBL ((char *) 0)
#define MM_NULLTAG ((char *) 0)
#define MM_NULLTXT ((char *) 0)
#define MM_NULLMC ((long int) 0)
#define MM_NULLSEV 0
#define MM_NOTOK -1 /* The function failed completely. */
#define MM_OK 0 /* The function succeeded. */
#define MM_NOMSG 1 /* The function was unable to generate a message on standard error, but otherwise succeeded. */
#define MM_NOCON 4 /* The function was unable to generate a console message, but otherwise succeeded. */

extern int fmtmsg(long int __classification, const char *__label, int __severity, const char *__text, const char *__action, const char *__tag);

13.4.11 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 *__pattern, const char *__name, int __flags);

13.4.12 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) (const char *__filename,
    const struct stat *__status, int __flag);
typedef int (*__ftw64_func_t) (const char *__filename,
    const struct stat64 *__status,
    int __flag);
typedef int (*__nftw_func_t) (const char *__filename,
    const struct stat *__status, int __flag,
    struct FTW *__info);
typedef int (*__nftw64_func_t) (const char *__filename,
    const struct stat64 *__status,
    int __flag,
    struct FTW *__info);
extern  int  ftw(const  char  *__dir,  __ftw_func_t  __func,  int
    __descriptors);
extern int ftw64(const char *__dir, __ftw64_func_t __func,
    int __descriptors);
extern int nftw(const char *__dir, __nftw_func_t __func, int
    __descriptors,
    int __flag);
extern int nftw64(const char *__dir, __nftw64_func_t __func,
    int __descriptors, int __flag);

13.4.13 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 ___argc, char *const ___argv[],
    const char *__shortopts,
    const struct option *__longopts, int *
    __longind);
extern int getopt_long_only(int ___argc, char *const ___argv[],
    const char *__shortopts,
    const struct option *__longopts,
    int *__longind);

13.4.14 glob.h

#define GLOB_ERR        (1<<0)
#define GLOB_MARK       (1<<1)
#define GLOB_BRACE      (1<<10)
#define GLOB_NOMAGIC    (1<<11)
#define GLOB_TILDE      (1<<12)
#define GLOB_ONLYDIR    (1<<13)
#define GLOB_TILDE_CHECK        (1<<14)
#define GLOB_NOSORT     (1<<2)
#define GLOB_DOOFFS     (1<<3)
#define GLOB_NOCHECK    (1<<4)
#define GLOB_APPEND     (1<<5)
#define GLOB_NOESCAPE   (1<<6)
#define GLOB_PERIOD     (1<<7)
#define GLOB_MAGCHAR    (1<<8)
#define GLOB_ALTDIRFUNC (1<<9)
```c
#define GLOB_NOSPACE    1
#define GLOB_ABORTED    2
#define GLOB_NOMATCH    3
#define GLOB_NOSYS      4

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

typedef struct {
  size_t gl_pathc;
  char **gl_pathv;
  size_t gl_offs;
  int gl_flags;
  void (*gl_closedir) (void *);
  struct dirent64 (*gl_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 *__pattern, int __flags,
               int (*__errfunc) (const char *, int), glob_t *__pglob);
extern int glob64(const char *__pattern, int __flags,
                  int (*__errfunc) (const char *, int),
                  glob64_t *__pglob);
extern void globfree(glob_t *
                       __pglob);
extern void globfree64(glob64_t *
                        __pglob);

13.4.15 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 int getgrent_r(struct group **__resultbuf, char *__buffer,
                       size_t __buflen, struct group **__result);
extern struct group *getgrgid(gid_t __gid);
extern int getgrgid_r(gid_t __gid, struct group *__resultbuf,
                       char *__buffer, size_t __buflen,
                       struct group **__result);
extern struct group *getgrnam(const char *__name);
extern int getgrnam_r(const char *__name, struct group *__resultbuf,
                       char *__buffer, size_t __buflen,
                       struct group **__result);
extern int getgrouplist(const char *__user, gid_t __group,
                        gid_t *__groups, int *__ngroups);
extern int initgroups(const char *__user, gid_t __group);
extern void setgrent(void);
extern void setgroups(size_t __n, const gid_t *__groups);
```

© 2008 Linux Foundation
13.4.16 iconv.h

typedef void *iconv_t;
extern size_t iconv(iconv_t __cd, char **__inbuf, size_t *__inbytesleft,
               char **__outbuf, size_t *__outbytesleft);
extern int iconv_close(iconv_t __cd);
extern iconv_t iconv_open(const char *__tocode, const char *__fromcode);

13.4.17 inttypes.h

typedef lldiv_t imaxdiv_t;
#define __PDP_ENDIAN 3412
#define PDP_ENDIAN __PDP_ENDIAN
extern intmax_t imaxabs(intmax_t __n);
extern imaxdiv_t imaxdiv(intmax_t __numer, intmax_t __denom);
extern intmax_t strtoimax(const char *__nptr, char **__endptr, int __base);
extern uintmax_t strtoumax(const char *__nptr, char **__endptr, int __base);
extern intmax_t wcstoimax(const wchar_t *__nptr, wchar_t **__endptr, int __base);
extern uintmax_t wcstoumax(const wchar_t * __nptr, wchar_t * __endptr, int __base);

13.4.18 langinfo.h

#define ABDAY_1 0x20000 /* Sun. */
#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 __item);

13.4.19 libgen.h

#define basename __xpg_basename
extern char *__xpg_basename(char *__path);
extern char *dirname(char *__path);

13.4.20 libintl.h

extern char *bind_textdomain_codeset(const char *__domainname, const char *__codeset);
extern char *bindtextdomain(const char *__domainname, const char *__dirname);
extern char *dcgettext(const char *__domainname, const char *__msgid, int __category);
extern char *dcngettext(const char *__domainname, const char *__msgid1, const char *__msgid2, unsigned long int __n, int __category);
extern char *dgettext(const char *__domainname, const char *__msgid);
extern char *dngettext(const char *__domainname, const char *__msgid1, const char *__msgid2, unsigned long int __n);
extern char *gettext(const char *__msgid);
extern char *ngettext(const char *__msgid1, const char *__msgid2, unsigned long int __n);
extern char *textdomain(const char *__domainname);

13.4.21 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_HOST_NAME_MAX 255
#define _POSIX_MAX_CANON 255
#define _POSIX_MAX_INPUT 255
#define _POSIX_SYMLINK_MAX 255
#define _POSIX2_EXPR_NEST_MAX 32
#define _POSIX_DELAYTIMER_MAX 32
#define _POSIX2_BC_BASE_MAX 99
#define _POSIX2_BC_SCALE_MAX 99
#define SSIZE_MAX LONG_MAX /* Maximum value of an object of type ssize_t */
#define BC_BASE_MAX _POSIX2_BC_BASE_MAX
#define BC_DIM_MAX _POSIX2_BC_DIM_MAX
#define BC_SCALE_MAX _POSIX2_BC_SCALE_MAX
#define BC_STRING_MAX _POSIX2_BC_STRING_MAX
#define EXPR_NEST_MAX _POSIX2_EXPR_NEST_MAX
#define _POSIX_FD_SETSIZE _POSIX_OPEN_MAX
#define _POSIX_HIWAT _POSIX_PIPE_BUF
#define MB_LEN_MAX 16
#define SCHAR_MIN (-128)
#define SCHAR_MAX 127
#define UCHAR_MAX 255
#define CHAR_BIT 8
#define SHRT_MIN (-32768)
#define SHRT_MAX 32767
#define USHRT_MAX 65535
#define INT_MIN (-INT_MAX-1)
#define INT_MAX 2147483647
#define UINT_MAX 4294967295U
#define LONG_MIN (-LONG_MAX-1L)
#define PTHREAD_KEYS_MAX 1024
#define PTHREAD_THREADS_MAX 16384
#define PTHREAD_DESTRUCTOR_ITERATIONS 4

13.4.22 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_TELEPHONE 10
#define LC_MEASUREMENT 11
#define LC_IDENTIFICATION 12
#define LC_TIME 2
#define LC_COLLATE 3
#define LC_MONETARY     4
#define LC_MESSAGES     5
#define LC_ALL          6
#define LC_PAPER        7
#define LC_NAME         8
#define LC_ADDRESS      9

struct __locale_struct {
    struct locale_data *__locales[13];
    const unsigned short *__ctype_b;
    const int *__ctypetolower;
    const int *__ctypetoupper;
    const char *__names[13];
};

typedef struct __locale_struct *__locale_t;
typedef struct __locale_struct *locale_t;

#define LC_ADDRESS_MASK (1 << LC_ADDRESS)
#define LC_COLLATE_MASK (1 << LC_COLLATE)
#define LC_IDENTIFICATION_MASK  (1 << LC_IDENTIFICATION)
#define LC_MEASUREMENT_MASK     (1 << LC_MEASUREMENT)
#define LC_MESSAGES_MASK        (1 << LC_MESSAGES)
#define LC_MONETARY_MASK        (1 << LC_MONETARY)
#define LC_NAME_MASK    (1 << LC_NAME)
#define LC_NUMERIC_MASK (1 << LC_NUMERIC)
#define LC_PAPER_MASK   (1 << LC_PAPER)
#define LC_TELEPHONE_MASK       (1 << LC_TELEPHONE)
#define LC_TIME_MASK    (1 << LC_TIME)
#define LC_CTYPE_MASK   (1<<LC_CTYPE)
#define LC_ALL_MASK     
    (LC_CTYPE_MASK|  LC_NUMERIC_MASK|  LC_TIME_MASK|
    LC_COLLATE_MASK| LC_MONETARY_MASK|
    LC_MESSAGES_MASK|  LC_PAPER_MASK|  LC_NAME_MASK|
    LC_ADDRESS_MASK| LC_TELEPHONE_MASK|
    LC_MEASUREMENT_MASK| LC_IDENTIFICATION_MASK)

extern locale_t duplocale(locale_t __dataset);
extern void freelocale(locale_t __dataset);
extern struct lconv *localeconv(void);
extern locale_t newlocale(int __category_mask, const char *__locale,
locale_t __base);
extern char *setlocale(int __category, const char *__locale);
extern locale_t uselocale(locale_t __dataset);

13.4.23 monetary.h

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

13.4.24 net/if.h

#define IF_NAMESIZE     16
#define IFF_UP  0x01            /* Interface is up. */
#define IFF_BROADCAST   0x02    /* Broadcast address valid. */
#define IFF_DEBUG       0x04    /* Turn on debugging. */
#define IFF_LOOPBACK    0x10    /* Is a loopback net. */
#define IFF_POINTOPOINT 0x10    /* Interface is point-to-point link. */
#define IFF_PROMISC     0x100   /* Receive all packets. */


```c
#define IFF_MULTICAST   0x1000 /* Supports multicast. */
#define IFF_NOTRAILERS  0x20    /* Avoid use of trailers. */
#define IFF_RUNNING     0x40    /* Resources allocated. */
#define IFF_NOARP       0x80    /* No address resolution protocol. */

struct if_nameindex {
    unsigned int if_index;  /* 1, 2, ... */
    char *if_name;          /* null terminated name: */
};

struct ifaddr {
    struct sockaddr ifa_addr; /* Address of interface. */
    union {
        struct sockaddr ifu_broadaddr;
        struct sockaddr ifu_dstaddr;
    } ifa_ifu;
    void *ifa_ifp;
    void *ifa_next;
};

#define ifr_name        ifr_ifrn.ifrn_name      /* interface name */
#define ifr_addr        ifr_ifru.ifru_addr      /* address */
#define ifr_broadaddr    ifr_ifru.ifru_broadaddr  /* broadcast address */
#define ifr_data        ifr_ifru.ifru_data      /* broadcast */
#define ifr_dstaddr     ifr_ifru.ifru_dstaddr   /* other end of p-p lnk */
#define ifr_flags       ifr_ifru.ifru_flags     /* flags */
#define ifr_hwaddr      ifr_ifru.ifru_hwaddr    /* interface name */
#define ifr_bandwidth   ifr_ifru.ifru_ivalue    /* link bandwidth index */
#define ifr_ifindex     ifr_ifru.ifru_ivalue    /* interface index */
#define ifr_metric      ifr_ifru.ifru_ivalue    /* metric */
#define ifr_qos         ifr_ifru.ifru_ivalue    /* queue length */
#define ifr_mtu         ifr_ifru.ifru_mtu       /* mtu */
#define ifr_netmask     ifr_ifru.ifru_netmask   /* interface netmask */
#define ifr_slave       ifr_ifru.ifru_slave     /* slave device */
#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_netmask;
        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;
};
```
#define ifc_buf ifc_ifcu.ifcu_buf       /* Buffer address. */
#define ifc_req ifc_ifcu.ifcu_req       /* Array of structures. */

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

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

13.4.25 netdb.h

#define h_errno (*__h_errno_location ()
#define NETDB_INTERNAL -1      /* See errno. */
#define NETDB_SUCCESS   0       /* No problem. */
#define HOST_NOT_FOUND  1       /* Authoritative Answer Host not found. */
#define IPPORT_RESERVED 1024
#define NI_MAXHOST 1025
#define TRY_AGAIN  2       /* Non-Authoritative Host not found, or SERVERFAIL. */
#define NO_RECOVERY  3       /* Non recoverable errors, FORMERR, REFUSED, NOTIMP. */
#define NI_MAXSERV  32
#define NO_DATA  4               /* Valid name, no data record of requested type. */
#define h_addr h_addr_list[0]
#define NO_ADDRESS NO_DATA /* No address, look for MX record. */

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 /* Socket address is intended for
`bind' */
#define AI_CANONNAME 0x0002 /* Request for canonical name */
#define AI_NUMERICHOST 0x0004 /* Don't use name resolution */
#define AI_V4MAPPED 0x0008 /* IPv4 mapped addresses are acceptable. */
#define AI_ALL 0x0010 /* Return IPv4 mapped and IPv6 addresses. */
#define AI_ADDRCONFIG 0x0020 /* Use configuration of this host to choose returned address type.. */
#define AI_NUMERICSERV 0x0400 /* Don't use name resolution */

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 /* Invalid value for `ai_flags' field. */
#define EAI_MEMORY -10 /* Memory allocation failure. */
#define EAI_SYSTEM -11 /* System error returned in 'errno'. */
#define EAI_NONAME -2 /* NAME or SERVICE is unknown. */
#define EAI_AGAIN -3 /* Temporary failure in name resolution. */
#define EAI_FAIL -4 /* Non-recoverable failure in name res. */
#define EAI_NODATA -5 /* No address associated with NAME. */
#define EAI_FAMILY -6 /* `ai_family' not supported. */
#define EAI_SOCKTYPE -7 /* `ai_family' not supported. */
#define EAI_SERVICE -8 /* SERVICE not supported for `ai_socktype'. */
#define EAI_ADDRFAMILY -9 /* Address family for NAME not supported. */

extern int *__h_errno_location(void);
extern void endprotoent(void);
extern void endservent(void);
extern void freeaddrinfo(struct addrinfo *__ai);
extern const char *gai_strerror(int __ecode);
extern int getaddrinfo(const char *__name, const char *__service, const struct addrinfo *__req, struct addrinfo **__pai);
extern struct hostent *gethostbyaddr(const void *__addr, socklen_t __len, int __type);
extern int gethostbyaddr_r(const void *__addr, socklen_t __len, int __type, struct hostent *__result_buf, char *__buf, size_t __buflen, struct hostent **__result, int *__h_errnop);
extern struct hostent *gethostbyname(const char *__name);
extern struct hostent *gethostbyname2(const char *__name, int __af);
extern int gethostbyname2_r(const char *__name, int __af,
    struct hostent *__result_buf, char *__buf,
    size_t __buflen, struct hostent **__result,
    int *__h_errno);
extern struct hostent *gethostbyname_r(const char *__name,
    struct hostent *__result_buf, char *__buf,
    size_t __buflen, struct hostent **__result,
    int *__h_errno);
extern struct protoent *getprotobyname(const char *__name);
extern int getprotobyname_r(const char *__name,
    struct protoent *__result_buf, char *__buf,
    size_t __buflen, struct protoent **__result,
    int *__h_errno);
extern struct protoent *getprotobynumber(int __proto);
extern int getprotobynumber_r(int __proto, struct protoent *
    __result_buf, char *__buf, size_t __buflen,
    struct protoent **__result);
extern struct protoent *getprotoent(void);
extern int getprotoent_r(struct protoent *__result_buf, char *
    __buf, size_t __buflen, struct protoent **__result);
extern void setprotoent(int __stay_open);
extern void setservent(int __stay_open);

13.4.26 netinet/icmp6.h

#define ICMP6_FILTER_WILLBLOCK(type,filterp)     ((((filterp)-
>icmp6_filt[(type) >> 5]) & (1 << ((type) & 31))) != 0)
#define ICMP6_FILTER_WILLPASS(type,filterp)     ((((filterp)-
>icmp6_filt[(type) >> 5]) & (1 << ((type) & 31))) == 0)
#define ICMP6_FILTER_SETPASS(type,filterp)      ((((filterp)-
>icmp6_filt[(type) >> 5]) &= ~(1 << ((type) & 31))))
#define ICMP6_FILTER_SETBLOCK(type,filterp)     ((((filterp)-
>icmp6_filt[(type) >> 5]) |= (1 << ((type) & 31))))
#define ICMP6_DST_UNREACH_NOROUTE       0

/*...*/
#define ICMP6_PARAMPROB_HEADER 0
#define ICMP6_TIME_EXCEED_TRANSIT 0
#define ICMP6_RR_FLAGS_PREVDONE 0x08
#define ICMP6_RR_FLAGS_SPECSITE 0x10
#define ICMP6_RR_PCOUSE_RAFLAGS_AUTO 0x10
#define ICMP6_RR_FLAGS_FORCEAPPLY 0x20
#define ICMP6_RR_PCOUSE_RAFLAGS_ONLINK 0x20
#define ND_OPT_PI_FLAG_RADDR 0x20
#define ND_RA_FLAG_HOME_AGENT 0x20
#define ICMP6_RR_FLAGS_REQRESULT 0x40
#define ND_OPT_PI_FLAG_AUTO 0x40
#define ND_RA_FLAG_OTHER 0x40
#define ICMP6_INFOMSG_MASK 0x80
#define ICMP6_RR_FLAGS_TEST 0x80
#define ND_OPT_PI_FLAG_ONLINK 0x80
#define ND_RA_FLAG_MANAGED 0x80
#define ICMP6_DST_UNREACH 1
#define ICMP6_FILTER 1
#define ICMP6_FILTER_BLOCK 1
#define ICMP6_PARAMPROB_NEXTHEDER 1
#define ICMP6_TIME_EXCEED_REASSEMBLY 1
#define ND_OPT_SOURCE_LINKADDR 1
#define RPM_PCO_ADD 1
#define ICMP6_ECHO_REQUEST 128
#define ICMP6_ECHO_REPLY 129
#define MLD_LISTENER_QUERY 130
#define MLD_LISTENER_REPORT 131
#define MLD_LISTENER_REDUCTION 132
#define ND_ROUTER_SOLICIT 133
#define ND_ROUTER_ADVERT 134
#define ND_NEIGHBOR_SOLICIT 135
#define ND_NEIGHBOR_ADVERT 136
#define ND_REDIRECT 137
#define ICMP6_ROUTER_RENUMBERING 138
#define ICMP6_DST_UNREACH_BEYONDSCOPE 2
#define ICMP6_FILTER_PASS 2
#define ICMP6 PACKET_TOO_BIG 2
#define ICMP6 PARAMPROB_OPTION 2
#define ND_OPT_TARGET_LINKADDR 2
#define RPM_PCO_CHANGE 2
#define ICMP6_DST_UNREACH_ADDR 3
#define ICMP6 FILTER BLOCKOTHERS 3
#define ICMP6 TIME_EXCEEDED 3
#define ICMP6 DST_UNREACH_NOPORT 4
#define ICMP6 FILTER BLOCKONLY 4
#define ICMP6_PARAM_PROB 4
#define ND_OPT_REDIRECTED_HEADER 4
#define ND_OPT_MTU 5
#define ND_OPT_RTR ADV INTERVAL 7
#define ND_OPT_HOME_AGENT INFO 8
#define icmp6_id icmp6_data16[0]
#define icmp6_maxdelay icmp6_data16[0]
#define icmp6_seq icmp6_data16[1]
#define icmp6_mtu icmp6_data32[0]
#define icmp6_pptr icmp6_data32[0]
#define icmp6 data16 icmp6_dataun.icmp6_un_data16
#define icmp6 data32 icmp6_dataun.icmp6_un_data32
#define icmp6 data8 icmp6_dataun.icmp6_un_data8
#define ICMP6 FILTER_SETPASSALL(filterp) memset (filterp, 0, sizeof (struct icmp6 filter));
#define ICMP6 FILTER_SETBLOCKALL(filterp) memset (filterp, 0xFF, sizeof (struct icmp6 filter));
#define mld_cksum mld_icmp6 hdr.icmp6_cksum
#define mld_code        mld_icmp6_hdr.icmp6_code
#define mld_maxdelay    mld_icmp6_hdr.icmp6_data16[0]
#define mld_reserved    mld_icmp6_hdr.icmp6_data16[1]
#define mld_type        mld_icmp6_hdr.icmp6_type
#define nd_na_cksum     nd_na_hdr.icmp6_cksum
#define nd_na_code      nd_na_hdr.icmp6_code
#define nd_na_flags_reserved nd_na_hdr.icmp6_data32[0]
#define nd_na_type      nd_na_hdr.icmp6_type
#define nd_ns_cksum     nd_ns_hdr.icmp6_cksum
#define nd_ns_code      nd_ns_hdr.icmp6_code
#define nd_ns_reserved  nd_ns_hdr.icmp6_data32[0]
#define nd_ns_type      nd_ns_hdr.icmp6_type
#define nd_ra_cksum     nd_ra_hdr.icmp6_cksum
#define nd_ra_code      nd_ra_hdr.icmp6_code
#define nd_ra_router_lifetime nd_ra_hdr.icmp6_data16[1]
#define nd_ra_flags_reserved nd_ra_hdr.icmp6_data8[0]
#define nd_ra_type      nd_ra_hdr.icmp6_type
#define nd_rd_cksum     nd_rd_hdr.icmp6_cksum
#define nd_rd_code      nd_rd_hdr.icmp6_code
#define nd_rd_reserved  nd_rd_hdr.icmp6_data32[0]
#define nd_rd_type      nd_rd_hdr.icmp6_type
#define nd_rs_cksum     nd_rs_hdr.icmp6_cksum
#define nd_rs_code      nd_rs_hdr.icmp6_code
#define nd_rs_reserved  nd_rs_hdr.icmp6_data32[0]
#define nd_rs_type      nd_rs_hdr.icmp6_type
#define rr_cksum        rr_hdr.icmp6_cksum
#define rr_code         rr_hdr.icmp6_code
#define rr_seqnum       rr_hdr.icmp6_data32[0]
#define rr_type         rr_hdr.icmp6_type

struct icmp6_filter {
    uint32_t icmp6_filt[8];
};

struct icmp6_hdr {
    uint8_t icmp6_type;
    uint8_t icmp6_code;
    uint16_t icmp6_cksum;
    union {
        uint32_t icmp6_un_data32[1];
        uint16_t icmp6_un_data16[2];
        uint8_t  icmp6_un_data8[4];
    } icmp6_dataun;
};

struct nd_router_solicit {
    struct icmp6_hdr nd_rs_hdr;
};

struct nd_router_advert {
    struct icmp6_hdr nd_ra_hdr;
    uint32_t nd_ra_reachable;
    uint32_t nd_ra_retransmit;
};

struct nd_neighbor_solicit {
    struct icmp6_hdr nd_ns_hdr;
    struct in6_addr nd_ns_target;
};

struct nd_neighbor_advert {
    struct icmp6_hdr nd_na_hdr;
    struct in6_addr nd_na_target;
};

struct nd_redirect {
    struct icmp6_hdr nd_rd_hdr;
    struct in6_addr nd_rd_target;
    struct in6_addr nd_rd_dst;
};

struct nd_opt_hdr {
struct nd_opt_prefix_info {
    uint8_t nd_opt_pi_type;
    uint8_t nd_opt_pi_len;
    uint8_t nd_opt_pi_prefix_len;
    uint8_t nd_opt_pi_flags_reserved;
    uint32_t nd_opt_pi_valid_time;
    uint32_t nd_opt_pi_preferred_time;
    uint32_t nd_opt_pi_reserved2;
    struct in6_addr nd_opt_pi_prefix;
};

struct nd_opt_rd_hdr {
    uint8_t nd_opt_rh_type;
    uint8_t nd_opt_rh_len;
    uint16_t nd_opt_rh_reserved1;
    uint32_t nd_opt_rh_reserved2;
};

struct nd_opt_mtu {
    uint8_t nd_opt_mtu_type;
    uint8_t nd_opt_mtu_len;
    uint16_t nd_opt_mtu_reserved;
    uint32_t nd_opt_mtu_mtu;
};

struct mld_hdr {
    struct icmp6_hdr mld_icmp6_hdr;
    struct in6_addr mld_addr;
};

struct icmp6_router_renum {
    struct icmp6_hdr rr_hdr;
    uint8_t rr_segnum;
    uint8_t rr_flags;
    uint16_t rr_maxdelay;
    uint32_t rr_reserved;
};

struct rr_pco_match {
    uint8_t rpm_code;
    uint8_t rpm_len;
    uint8_t rpm_ordinal;
    uint8_t rpm_matchlen;
    uint8_t rpm_minlen;
    uint8_t rpm_maxlen;
    uint16_t rpm_reserved;
    struct in6_addr rpm_prefix;
};

struct rr_pco_use {
    uint8_t rpu_uselen;
    uint8_t rpu_keeplen;
    uint8_t rpu_ramask;
    uint8_t rpu_raflags;
    uint32_t rpu_vltime;
    uint32_t rpu_pltime;
    uint32_t rpu_flags;
    struct in6_addr rpu_prefix;
};

struct rr_result {
    uint16_t rrr_flags;
    uint8_t rrr_ordinal;
    uint8_t rrr_matchedlen;
    uint32_t rrr_ifid;
    struct in6_addr rrr_prefix;
};

struct nd_opt_adv_interval {
    uint8_t nd_opt_adv_interval_type;
    uint8_t nd_opt_adv_interval_len;
}
typedef uint16_t nd_opt_adv_interval_reserved;
typedef uint32_t nd_opt_adv_interval_ival;
}

struct nd_opt_home_agent_info {
    uint8_t nd_opt_home_agent_info_type;
    uint8_t nd_opt_home_agent_info_len;
    uint16_t nd_opt_home_agent_info_reserved;
    int16_t nd_opt_home_agent_info_preference;
    uint16_t nd_opt_home_agent_info_lifetime;
};

13.4.27 netinet/igmp.h

#define IGMP_MEMBERSHIP_QUERY 0x11
#define IGMP_V1_MEMBERSHIP_REPORT 0x12
#define IGMP_DVMRP 0x13
#define IGMP_PIM 0x14
#define IGMP_TRACE 0x15
#define IGMP_V2_MEMBERSHIP_REPORT 0x16
#define IGMP_V2_LEAVE_GROUP 0x17
#define IGMP_MTRACE_RESP 0x1e
#define IGMP_MTRACE 0x1f
#define IGMP_DELAYING_MEMBER 1
#define IGMP_v1_ROUTER 1
#define IGMP_MAX_HOST_REPORT_DELAY 10
#define IGMP_TIMER_SCALE 10
#define IGMP_IDLE_MEMBER 2
#define IGMP_v2_ROUTER 2
#define IGMP_LAZY_MEMBER 3
#define IGMP_SLEEPING_MEMBER 4
#define IGMP_AWAKENING_MEMBER 5
#define IGMP_MINLEN 8
#define IGMP_HOST_MEMBERSHIP_QUERY IGMP_MEMBERSHIP_QUERY
#define IGMP_HOST_MEMBERSHIP_REPORT IGMP_V1_MEMBERSHIP_REPORT
#define IGMP_HOST_LEAVE_MESSAGE IGMP_V2_LEAVE_GROUP
#define IGMP_HOST_NEW_MEMBERSHIP_REPORT IGMP_V2_MEMBERSHIP_REPORT

struct igmp {
    u_int8_t igmp_type;
    u_int8_t igmp_code;
    u_int16_t igmp_cksum;
    struct in_addr igmp_group;
};

13.4.28 netinet/in.h

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

typedef uint16_t in_port_t;

typedef uint32_t in_addr_t;

#define INADDR_NONE ((in_addr_t) 0xffffffff)
#define INADDR_BROADCAST        (0xffffffff)
#define INADDR_ANY      0
#define INADDR_LOOPBACK 0x7f000001 /* 127.0.0.1 */

#define s6_addr16 in6_u.u6_addr16
#define s6_addr32 in6_u.u6_addr32
#define s6_addr in6_u.u6_addr8

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 IN_MULTICAST(a)  ((((in_addr_t) (a))&0xf0000000)==0xe0000000)
#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 IN6_IS_ADDR_LINKLOCAL(a)        ((((const uint32_t *) (a))[0] & htonl (0xffc00000)) == htonl (0xfe800000))
#define IN6_IS_ADDR_SITELOCAL(a)        ((((const uint32_t *) (a))[0] & htonl (0xffc00000)) == htonl (0xfec00000))
#define IN6_ARE_ADDR_EQUAL(a,b) ((((const uint32_t *) (a))[0] == ((const uint32_t *) (b))[0]) && (((const uint32_t *) (a))[1] == ((const uint32_t *) (b))[1]) && (((const uint32_t *) (a))[2] == ((const uint32_t *) (b))[2]) && (((const uint32_t *) (a))[3] == ((const uint32_t *) (b))[3]))
#define IN6_IS_ADDR_V4COMPAT(a) ((((const uint32_t *) (a))[0] == 0) && (((const uint32_t *) (a))[1] == 0) && (((const uint32_t *) (a))[2] == 0) && (ntohl ((const uint32_t *) (a))[3]) > 1))
#define IN6_IS_ADDR_V4MAPPED(a) ((((const uint32_t *) (a))[0] == 0) && (((const uint32_t *) (a))[1] == 0) && (((const uint32_t *) (a))[2] == htonl (0xffff)))
#define IN6_IS_ADDR_UNSPECIFIED(a)      (((const uint32_t *) (a))[0] == 0xff)
#define IN6_IS_ADDR_MC_NODELOCAL(a)     ((IN6_IS_ADDR_MULTICAST(a) && ((((const uint8_t *) (a))[1] & 0xf) == 0x1))
#define IN6_IS_ADDR_MC_LINKLOCAL(a)     ((IN6_IS_ADDR_MULTICAST(a) && ((((const uint8_t *) (a))[1] & 0xf) == 0x2))
#define IN6_IS_ADDR_MC_SITELOCAL(a)     ((IN6_IS_ADDR_MULTICAST(a) && ((((const uint8_t *) (a))[1] & 0xf) == 0x5))
#define IN6_IS_ADDR_MC_ORGLOCAL(a)      ((IN6_IS_ADDR_MULTICAST(a) && ((((const uint8_t *) (a))[1] & 0xf) == 0x8))
#define IN6_IS_ADDR_MC_GLOBAL(a)        ((IN6_IS_ADDR_MULTICAST(a) && ((((const uint8_t *) (a))[1] & 0xf) == 0xe))

#define INET6_ADDRSTRLEN 46
struct sockaddr_in6 {
    unsigned short sin6_family; /* AF_INET6 */
    uint16_t sin6_port;         /* Transport layer port # */
    uint32_t sin6_flowinfo;     /* IPv6 flow information */
    struct in6_addr sin6_addr;  /* IPv6 address */
    uint32_t sin6_scope_id;     /* scope id (new in RFC2553) */
};

#define SOL_IP    0
#define IP_TOS    1               /* IP type of service and precedence */
#define IPV6_UNICAST_HOPS 16
#define IPV6_MULTICAST_IF 17
#define IPV6_MULTICAST_HOPS 18
#define IPV6_MULTICAST_LOOP 19
#define IP_TTL     2               /* IP time to live */
#define IPV6_JOIN_GROUP 20
#define IPV6_LEAVE_GROUP 21
#define IPV6_V6ONLY 26
#define IP_MULTICAST_IF 32         /* set/get IP multicast i/f */
#define IP_MULTICAST_TTL 33        /* set/get IP multicast ttl */
#define IP_MULTICAST_LOOP 34       /* set/get IP multicast loopback */
#define IP_ADD_MEMBERSHIP 35       /* add an IP group membership */
#define IP_DROP_MEMBERSHIP 36      /* drop an IP group membership */
#define IP_OPTIONS  4               /* IP per-packet options */
#define IPV6_ADD_MEMBERSHIP IPV6_JOIN_GROUP
#define IPV6_DROP_MEMBERSHIP IPV6_LEAVE_GROUP

struct ipv6_mreq {
    struct in6_addr ipv6mr_multiaddr; /* IPv6 multicast address of group */
    int ipv6mr_interface; /* local IPv6 address of interface */
};

struct ip_mreq {
    struct in_addr imr_multiaddr;       /* IP multicast address of group */
    struct in_addr imr_interface;       /* local IP address of interface */
};

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

13.4.29 netinet/in_systm.h

typedef u_int16_t n_short;
typedef u_int32_t n_long;
typedef u_int32_t n_time;

13.4.30 netinet/ip.h

#define IPOPT_CLASS(o)  ((o) & IPOPT_CLASS_MASK)
#define IPOPT_COPIED(o) ((o) & IPOPT_COPY)
#define IPOPT_NUMBER(o) ((o) & IPOPT_NUMBER_MASK)
#define IPOPT_EOL       0
#define IPOPT_OPTVAL    0
#define IPOPT_TS_TSONLY 0
#define IPOPT_CONTROL 0x00
#define IPOPT_SECUR_UNCLASS 0x0000
#define IPOPT_NUMBER_MASK 0x1f
#define IP_OFFMASK 0x1fff
#define IPOPT_RESERVED1 0x20
#define IP_MF 0x2000
#define IPOPT_DEBMEAS 0x40
#define IP_DF 0x4000
#define IPOPT_CLASS_MASK 0x60
#define IPOPT_RESERVED2 0x60
#define IPOPT_SECUR_TOPSECRET 0x6bc5
#define IPOPT_SECUR_EFTO 0x789a
#define IPOPT_COPY 0x80
#define IP_RF 0x8000
#define IPOPT_SECUR_RESTRICT 0xaf13
#define IPOPT_SECUR_MMMM 0xbc4d
#define IPOPT_SECUR_SECRET 0xd788
#define IPOPT_SECUR_CONFIDENT 0xf135
#define IPOPT_NOP 1
#define IPOPT_OLEN 1
#define IPOPT_TS_TSANDADDR 1
#define IPTTLDEC 1
#define IPOPT_SECURITY 130
#define IPOPT_LSRR 131
#define IPOPT_SATID 136
#define IPOPT_SSRR 137
#define IPOPT_RA 148
#define IPOPT_OFFSET 2
#define MAXTTL 255
#define IPOPT_TS_PRESPEC 3
#define IPOPT_MINOFF 4
#define MAX_IPOPTLEN 40
#define IP_MSS 576
#define IPFRAGTTL 60
#define IPDEFTTL 64
#define IP_MAXPACKET 65535
#define IPOPT_TS 68
#define IPOPT_RR 7
#define IPOPT_MEASUREMENT IPOPT_DEBMEAS
#define IPOPT_END IPOPT_EOL
#define IPOPT_NOOP IPOPT_NOP
#define IPOPT_SID IPOPT_SATID
#define IPOPT_SEC IPOPT_SECURITY
#define IPOPT_TIMESTAMP IPOPT_TS

#define IPTOS_TOS(tos) ((tos) & IPTOS_TOS_MASK)
#define IPTOS_MINCOST IPTOS_LOWCOST
#define IPTOS_PREC(tos) ((tos) & IPTOS_PREC_MASK)
#define IPTOS_TOS_MASK 0x1e

#define IP6OPT_TYPE(o) ((o) & 0xc0)
#define IP6OPT_PAD1 0
#define IP6OPT_TYPE_SKIP 0x00
#define IP6OPT_TUNNEL_LIMIT 0x04
#define IP6OPT_ROUTER_ALERT 0x05
#define IP6OPT_RELABEL 0x06
#define IP6OPT_REGISTRATION 0x07
#define IP6OPT_ND6 0x08
#define IP6OPT_MINCOST IP6OPT_LOWEST
#define IP6OPT_LSRR 0x09
#define IP6OPT_SATID 0x0a
#define IP6OPT_SSR 0x0b
#define IP6OPT_RA 0x0c
#define IP6OPT_OFFSET 0x0d
#define IP6OPT_MEASUREMENT IP6OPT_DEBMEAS
#define IP6OPT_END IP6OPT_EOL
#define IP6OPT_NOOP IP6OPT_NOP
#define IP6OPT_SID IP6OPT_SATID
#define IP6OPT_SEC IP6OPT_SECURITY
#define IP6OPT_TIMESTAMP IP6OPT_TS

#define IP6OPT_TOS(tos) ((tos) & IP6OPT_TOS_MASK)
#define IP6OPT_MINCOST IP6OPT_LOWEST
#define IP6OPT_PREC(tos) ((tos) & IP6OPT_PREC_MASK)
#define IP6OPT_TOS_MASK 0x1e
# define IP6OPT_TYPE_MUTABLE     0x20
# define IP6OPT_TYPE_DISCARD     0x40
# define IP6OPT_TYPE_FORCEICMP   0x80
# define IP6OPT_TYPE_ICMP        0xc0
# define IP6OPT_JUMBO    0xc2
# define IP6OPT_NSAP_ADDR        0xc3
# define IP6OPT_PADN           1
# define IP6OPT_JUMBO_LEN        6
#define ip6_flow        ip6_ctlun.ip6_un1.ip6_un1_flow
#define ip6_hlim        ip6_ctlun.ip6_un1.ip6_un1_hlim
#define ip6_hops        ip6_ctlun.ip6_un1.ip6_un1_hlim
#define ip6_nxt ip6_ctlun.ip6_un1.ip6_un1_nxt
#define ip6_plen        ip6_ctlun.ip6_un1.ip6_un1_plen
#define ip6_vfc ip6_ctlun.ip6_un2_vfc

struct ip6_hdrctl {
    uint32_t ip6_un1_flow;
    uint16_t ip6_un1_plen;
    uint8_t ip6_un1_nxt;
    uint8_t ip6_un1_hlim;
};

struct ip6_hdr {
    struct in6_addr ip6_src;
    struct in6_addr ip6_dst;
};

struct ip6_ext {
    uint8_t ip6e_nxt;
    uint8_t ip6e_len;
};

struct ip6_hbh {
    uint8_t ip6h_nxt;
    uint8_t ip6h_len;
};

struct ip6_dest {
    uint8_t ip6d_nxt;
    uint8_t ip6d_len;
};

struct ip6_rthdr {
    uint8_t ip6r_nxt;
    uint8_t ip6r_len;
    uint8_t ip6r_type;
    uint8_t ip6r_segleft;
};

struct ip6_frag {
    uint8_t ip6f_nxt;
    uint8_t ip6f_reserved;
    uint16_t ip6f_offlg;
    uint32_t ip6f_ident;
};

struct ip6_opt {
    uint8_t ip6o_type;
    uint8_t ip6o_len;
};

struct ip6_opt_jumbo {
    uint8_t ip6oj_type;
    uint8_t ip6oj_len;
    uint8_t ip6oj_jumbo_len[4];
};

struct ip6_opt_nsap {
    uint8_t ip6on_type;
    uint8_t ip6on_len;
    uint8_t ip6on_src_nsap_len;
    uint8_t ip6on_dst_nsap_len;
};

struct ip6_opt_tunnel {
    uint8_t ip6ot_type;
}
13 Base Libraries

ISO/IEC 23360 Part 1:2008(E)

```c
uint8_t ip6ot_len;
uint8_t ip6ot_encap_limit;
};
struct ip6_opt_router {
    uint8_t ip6or_type;
    uint8_t ip6or_len;
    uint8_t ip6or_value[2];
};

13.4.32 netinet/ip_icmp.h

#define ICMP_INFOTYPE(type)     ((type) == ICMP_ECHOREPLY ||
                              (type) == ICMP_ECHO ||
                              (type) == ICMP_ROUTERADVERT ||
                              (type) == ICMP_ROUTERSOLICIT ||
                              (type) == ICMP_TSTAMP ||
                              (type) == ICMP_TSTAMPREPLY ||
                              (type) == ICMP_IREQ ||
                              (type) == ICMP_IREQREPLY ||
                              (type) == ICMP_MASKREQ ||
                              (type) == ICMP_MASKREPLY)
#define ICMP_ADVLEN(p)  (8 + ((p)->icmp_ip.ip_hl << 2) + 8)
#define ICMP_TSLEN      (8 + 3 * sizeof (n_time))
#define ICMP_ADVLENMIN  (8 + sizeof (struct ip) + 8)
#define ICMP_ECHOREPLY  0
#define ICMP_EXC_TTL    0
#define ICMP_NET_UNREACH 0
#define ICMP_REDIRECT_NET 0
#define ICMP_REDIR_NET  0
#define ICMP_TIMXCEED_INTRANS 0
#define ICMP_UNREACH_NET 0
#define ICMP_EXC_FRAGTIME 1
#define ICMP_HOST_UNREACH 1
#define ICMP_PARAMPROB_OPTABSENT 1
#define ICMP_REDIRECT_HOST 1
#define ICMP_REDIR_HOST 1
#define ICMP_TIMXCEED_REASS 1
#define ICMP_UNREACH_HOST 1
#define ICMP_HOST_AND 10
#define ICMP_ROUTERSOLICIT 10
#define ICMP_UNREACH_HOST_PROHIB 10
#define ICMP_NET_UNR_TOS  11
#define ICMP_TIME_EXCEEDED 11
#define ICMP_TIMXCEED  11
#define ICMP_UNREACH_TOSNET 11
#define ICMP_HOST_UNR_TOS 12
#define ICMP_MASKLEN  12
#define ICMP_PARAMETERPROB 12
#define ICMPgetParam  12
#define ICMP_UNREACH_TOSHOST 12
#define ICMP_PKT_FILTERED 13
#define ICMP_TIMESTAMP  13
#define ICMP_TSTAMP  13
#define ICMP_UNREACH_FILTER_PROHIB 13
#define ICMP_PREC_VIOLATION 14
#define ICMP_TIMESTAMPREPLY 14
#define ICMP_TSTAMPREPLY 14
#define ICMP_UNREACH_HOST_PRECEDENCE 14
#define ICMP_INFO_REQUEST 15
#define ICMP_INFO_REPLY  16
#define ICMP_ADDRESS  17
#define ICMP_MASKREQ  17
#define ICMP_ADDRESSREPLY 18
```
ISO/IEC 23360 Part 1:2008(E)

#define ICMP_MASKREPLY 18
#define ICMP_MAXTYPE 18
#define NR_ICMP_TYPES 18
#define ICMP_PROT_UNREACH 2
#define ICMP_REDIRECT_TOSNET 2
#define ICMPRedirect_NETTOS 2
#define ICMP_REDIR_NETTOS 2
#define ICMP_REDIR_NETTOS 2
#define ICMP_UNREACH_PROTOCOL 2
#define ICMP_DEST_UNREACH 3
#define ICMP_PORT_UNREACH 3
#define ICMP_REDIRECT_TOSHOST 3
#define ICMP_REDIRECT_HOSSTOS 3
#define ICMP_REDIRECT_HOSTTOS 3
#define ICMP_REDIRECT_HOSTTOS 3
#define ICMP_UNREACH 3
#define ICMP_UNREACH_PORT 3
#define ICMP_FRAG_NEEDED 4
#define ICMP_SOURCEQUENCH 4
#define ICMP_SOURCE_QUENCH 4
#define ICMP_UNREACH_NEEDFRAG 4
#define ICMP_REDIRECT 5
#define ICMP_REDIRECT 5
#define ICMP_REDIRECT 5
#define ICMP_REDIRECT 5
#define ICMP_REDIRECT 5
#define ICMP_REDIRECT 5
#define ICMP_SR_FAILED 5
#define ICMP_SR_FAILED 5
#define ICMP_SR_FAILED 5
#define ICMP_SR_FAILED 5
#define ICMP_SR_FAILED 5
#define ICMP_SR_FAILED 5
#define ICMP_NET_UNKNOWN 6
#define ICMP_NET_UNKNOWN 6
#define ICMP_UNREACH_NET_UNKNOWN 6
#define ICMP_HOST_UNKNOWN 7
#define ICMP_HOST_UNKNOWN 7
#define ICMP_UNREACH_HOST_UNKNOWN 7
#define ICMP_ECHO 8
#define ICMP_ECHO 8
#define ICMP_ECHO 8
#define ICMP_ECHO 8
#define ICMP_ECHO 8
#define ICMP_ECHO 8
#define ICMP_MINLEN 8
#define ICMP_UNREACH_ISOLATED 8
#define ICMP_UNREACH_ISOLATED 8
#define ICMP_UNREACH_ISOLATED 8
#define ICMP_NET_ANO 9
#define ICMP_NET_ANO 9
#define ICMP_ROUTERADVERT 9
#define ICMP_ROUTERADVERT 9
#define ICMP_UNREACH_NET_PROHIB 9
#define icmp_data icmp_dun.id_data
#define icmp_ip icmp_dun.id_ip.idi_ip
#define icmp_mask icmp_dun.id_mask
#define icmp_radv icmp_dun.id_radv
#define icmp_otime icmp_dun.id_ts.its_otime
#define icmp_rtime icmp_dun.id_ts.its_rtime
#define icmp_ttime icmp_dun.id_ts.its_ttime
#define icmp_gwaddr icmp_dun.id_gwaddr
#define icmp_id icmp_hun.ih_idseq.icd_id
#define icmp_seq icmp_hun.ih_idseq.icd_seq
#define icmp_nextmtu icmp_hun.ih_pmtu.ipm_nextmtu
#define icmp_pmvoid icmp_hun.ih_pmtu.ipm_void
#define icmp_pptr icmp_hun.ih_pptr
#define icmp_lifetime icmp_hun.ih_rtradv.irt_lifetime
#define icmp_num_addr icmp_hun.ih_rtradv.irt_num_addr
#define icmp_wpa icmp_hun.ih_rtradv.irt_wpa
#define icmp_void icmp_hun.ih_void

struct icmphdr {
    u_int8_t type;
    u_int8_t code;
    u_int16_t checksum;
    union {
        struct {
            u_int16_t id;
            u_int16_t sequence;
        } echo;
        u_int32_t gateway;
        struct {
            u_int16_t unused;
            u_int16_t mtu;
        } frag;
    } un;
};
struct icmp_ra_addr {
13 Base Libraries

ISO/IEC 23360 Part 1:2008(E)

u_int32_t ira_addr;
u_int32_t ira_preference;
};
struct ih_idseq {
    u_int16_t icd_id;
    u_int16_t icd_seq;
};
struct ih_pmtu {
    u_int16_t ipm_void;
    u_int16_t ipm_nextmtu;
};
struct ih_rtradv {
    u_int8_t irt_num_addrs;
    u_int8_t irt_wpa;
    u_int16_t irt_lifetime;
};
struct icmp {
    u_int8_t icmp_type;
    u_int8_t icmp_code;
    u_int16_t icmp_cksum;
    union {
        u_int16_t ih_pptr;
        struct in_addr ih_gwaddr;
        struct ih_idseq ih_idseq;
        u_int32_t ih_void;
        struct ih_pmtu ih_pmtu;
        struct ih_rtradv ih_rtradv;
    } icmp_hun;
    union {
        struct {
            u_int32_t its_otime;
            u_int32_t its_rtime;
            u_int32_t its_ttime;
        } id_ts;
        struct {
            struct ip idi_ip;
        } id_ip;
        struct icmp_ra_addr id_radv;
        u_int32_t id_mask;
        u_int8_t id_data[1];
    } icmp_dun;
};

13.4.33 netinet/tcp.h

#define TCPOLEN_TSTAMP_APPA   (TCPOLEN_TIMESTAMP+2)
#define TCPOPT_TSTAMP_HDR     (TCPOPT_NOP<<24|TCPOPT_NOP<<16|
                           TCPOPT_TIMESTAMP<<8|TCPOLEN_TIMESTAMP)
#define TCP_EOPL         0
#define TCPI_OPT_TIMESTAMPS 1
#define TCPOPT_NOP       1
#define TCP_NODELAY     1
#define TCPOLEN_TIMESTAMP 10
#define TCP_WINDOW_CLAMP 10
#define TCP_INFO        11
#define TCP_QUICKACK    12
#define TCP_CONGESTION  13
#define TCP_MAX_WINSHIFT 14
#define TCPI_OPT_SACK   2
#define TCPOLEN_SACK_PERMITTED 2
#define TCPOPT_MAXSEG   2
#define TCP_MAXSEG      2
#define TCPOLEN_WINDOW  3
#define TCPOPT_WINDOW   3
#define TCP_CORK        3
#define TCPI_OPT_WSCALE 4
#define TCPOLEN_MAXSEG  4
#define TCPOPT_SACK_PERMITTED   4
#define TCP_KEEPIDLE    4
#define TCPOPT_SACK      5
#define TCP_KEEPINTVL   5
#define TCP_MSS 512
#define SOL_TCP 6
#define TCP_KEEPCNT     6
#define TCP_MAXWIN 65535
#define TCP_SYNCTIME 7
#define TCPI_OPT_ECN    8
#define TCPOPT_TIMESTAMP        8
#define TCP_LINGER2     8
#define TCP_DEFER_ACCEPT        9

enum tcp_ca_state {
    TCP_CA_Open,
    TCP_CA_Disorder,
    TCP_CA_CWR,
    TCP_CA_Recovery,
    TCP_CA_Loss
};

struct tcp_info {
    uint8_t tcpi_state;
    uint8_t tcpi_ca_state;
    uint8_t tcpi_retransmits;
    uint8_t tcpi_probes;
    uint8_t tcpi_backoff;
    uint8_t tcpi_options;
    uint8_t tcpi_snd_wscale:4;
    uint8_t tcpi_rcv_wscale:4;
    uint32_t tcpi_rto;
    uint32_t tcpi_ato;
    uint32_t tcpi_snd_mss;
    uint32_t tcpi_rcv_mss;
    uint32_t tcpi_unacked;
    uint32_t tcpi_sacked;
    uint32_t tcpi_lost;
    uint32_t tcpi_retrans;
    uint32_t tcpi_fackets;
    uint32_t tcpi_last_data_sent;
    uint32_t tcpi_last_ack_sent;
    uint32_t tcpi_last_data_recv;
    uint32_t tcpi_last_ack_recv;
    uint32_t tcpi_pmtu;
    uint32_t tcpi_rcv_ssthresh;
    uint32_t tcpi_snd_ssthresh;
    uint32_t tcpi_snd_cwnd;
    uint32_t tcpi_advmss;
    uint32_t tcpi_reordering;
};

enum {
    TCP_ESTABLISHED = 1,
    TCP_SYN_SENT = 2,
    TCP_SYN_RECV = 3,
    TCP_FIN_WAIT1 = 4,
    TCP_FIN_WAIT2 = 5,
    TCP_TIME_WAIT = 6,
    TCP_CLOSE = 7,
    TCP_CLOSE_WAIT = 8,
    TCP_LAST_ACK = 9,
    TCP_LISTEN = 10,
13.4.34 netinet/udp.h

#define SOL_UDP 17
struct udphdr {
    u_int16_t source;
    u_int16_t dest;
    u_int16_t len;
    u_int16_t check;
};

13.4.35 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 __catalog);
extern char *catgets(nl_catd __catalog, int __set, int __number,
                     const char *__string);
extern nl_catd catopen(const char *__cat_name, int __flag);

13.4.36 poll.h

extern int poll(struct pollfd *__fds, nfds_t __nfds, int __timeout);

13.4.37 pty.h

extern int forkpty(int *__amaster, char *__name, struct termios *__term,
                    struct winsize *__winp);
extern int openpty(int *__amaster, int *__slave, char *__name,
                    struct termios *__term, struct winsize *__winp);

13.4.38 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 int getpwent_r(struct passwd *__resultbuf, char *__buffer,
                       size_t __buflen, struct passwd **__result);
extern struct passwd *getpwnam(const char *__name);
extern int getpwnam_r(const char *__name, struct passwd
* __resultbuf, char *__buffer, size_t __buflen, 
  struct passwd **__result); 
extern struct passwd *getpwuid(uid_t __uid); 
extern int getpwuid_r(uid_t __uid, struct passwd *__resultbuf, 
  char *__buffer, size_t __buflen, struct passwd **__result); 
extern void setpwent(void); 

13.4.39 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 reg_ascallocated: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_EPAREN = 8, 
  REG_EBRACE = 9, 
  REG_BADBR = 10, 
  REG_ERANGE = 11, 
  REG_ESPACE = 12, 
  REG_EBRACE = 13, 
  REG_EERN = 14, 
  REG_ESIZE = 15, 
  REG_ERPAREN = 16 
}
extern reg_errcode_t;
extern int regcomp(regex_t * __preg, const char *__pattern, int __cflags);
extern size_t regerror(int __errcode, const regex_t * __preg, char *__errbuf, size_t __errbuf_size);
extern int regexec(const regex_t * __preg, const char *__string, size_t __nmatch, regmatch_t __pmatch[], int __eflags);
extern void regfree(regex_t * __preg);

13.4.40 rpc/auth.h

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

enum auth_stat {
    AUTH_OK = 0,
    AUTH_BADCRED = 1,  /* bogus credentials (seal broken) */
    AUTH_REJECTEDCRED = 2, /* client should begin new session */
    AUTH_BADVERF = 3,  /* bogus verifier (seal broken) */
    AUTH_REJECTEDVERF = 4, /* verifier expired or was replayed */
    AUTH_TOOWEAK = 5,  /* Rpc calls return an enum clnt_stat. */
    AUTH_INVALIDRESP = 6, /* bogus response verifier */
    AUTH_FAILED = 7,   /* some unknown reason */
};

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

struct opaque_auth {
    enum_t oa_flavor;  /* flavor of auth */
    caddr_t oa_base;   /* address of more auth stuff */
    u_int oa_length;  /* not to exceed MAX_AUTH_BYTES */
};

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 (*ahMarshal) (struct AUTH *, XDR *); /* nextverf & serialize */
    int (*ah_validate) (struct AUTH *, struct opaque_auth *); /* validate verifier */
    int (*ah_refresh) (struct AUTH *); /* refresh credentials */
    void (*ah_destroy) (struct AUTH *); /* Rpc calls return an enum clnt_stat. */
};
extern struct AUTH *authnone_create(void);
extern int key_decryptsession(char *, union des_block *);
extern bool_t xdr_opaque_auth(XDR *, struct opaque_auth *);

13.4.41 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)     /* By convention, procedure 0 takes null arguments and returns */
#define CLSET_TIMEOUT   1      /* set timeout (timeval) */
#define CLGET_XID        10     /* Get xid */
#define CLSET_XID        11     /* Set xid */
#define CLGET_VERS       12     /* Get version number */
#define CLSET_VERS       13     /* Set version number */
#define CLGET_PROG       14     /* Get program number */
#define CLSET_PROG       15     /* Set program number */
#define CLGET_TIMEOUT   2      /* get timeout (timeval) */
#define CLGET_SERVER_ADDR  3     /* get server's address (sockaddr) */
#define CLGET_RETRY_TIMEOUT  4  /* set retry timeout (timeval) */
#define CLGET_RETRY_TIMEOUT  5  /* get retry timeout (timeval) */
#define CLGET_FD        6      /* get connections file descriptor */
#define CLGET_SVC_ADDR  7      /* get server's address (netbuf) */
#define CLSET_FD_CLOSE  8      /* close fd while clnt_destroy */
#define CLSET_FD_NCLOSE 9      /* Do not close fd while clnt_destroy */
#define clnt_call(rh, proc, xargs, argsp, xres, resp, secs)     
  (*((rh)->cl_ops->cl_call)(rh, proc, xargs, argsp, xres, resp, secs))

enum clnt_stat {
  RPC_SUCCESS = 0,     /* call succeeded */
  RPC_CANTENCODERARGS = 1, /* can't encode arguments */
  RPC_CANTDECODEARGS = 2, /* can't decode results */
  RPC_CANTSEND = 3,     /* failure in sending call */
  RPC_CANTRECV = 4,     /* failure in receiving result */
  RPC_TIMEDOUT = 5,     /* call timed out */
  RPC_VERSMISMATCH = 6, /* rpc versions not compatible */
  RPC_AUTHERROR = 7,     /* authentication error */
  RPC_PROGRAMAVAILABLE = 8, /* program not available */
  RPC_PROGRAMVERSIONMISMATCH = 9, /* program version mismatched */
  RPC_PROCUNAVAILABLE = 10, /* procedure unavailable */
  RPC_CANTDECODEARGS = 11, /* decode arguments error */
  RPC_SYSTEMERROR = 12,  /* generic "other problem" */
  RPC_NOBROADCAST = 21,  /* Broadcasting not supported */
  RPC_UNKNOWNHOST = 13,  /* unknown host name */
  RPC_UNKNOWNPROTOCOL = 17, /* unknown protocol */
  RPC_REMOTEADDRESS = 19, /* Remote address unknown */
  RPC_PORTMAPPERFAILED = 14, /* portmapper failed in its call */
  RPC_PROGRAMNOTREGISTERED = 15, /* remote program is not registered */
  RPC_N2AXLATEFAILURE = 22, /* Name to addr translation
failed */
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 int callrpc(const char *__host, const u_long __prognum,
    const u_long __versnum, const u_long __procnum,
    const xdrproc_t __inproc, const char *__in,
    const xdrproc_t __outproc, char *__out);

extern struct CLIENT *clnt_create(const char *__host, const
    u_long __prog, const u_long __vers, const char *
    __prot);

extern void clnt_pcreateerror(const char *__msg);
extern void clnt_perrno(enum clnt_stat __num);
extern void clnt_perror(struct CLIENT *__clnt, const char *
    __msg);
extern char *clnt_spcreateerror(const char *__msg);
extern char *clnt_sperrno(enum clnt_stat __num);
extern char *clnt_sperror(struct CLIENT *__clnt, const char *
    __msg);
extern struct CLIENT *clntraw_create(u_long __prog, u_long
    __vers);
extern struct CLIENT *clnttcp_create(struct sockaddr_in *__raddr,
    u_long __prog, u_long __version,
    int *__sockp, u_int __sendsz,
    u_int __recvysz);
extern struct CLIENT *clntudp_bufcreate(struct sockaddr_in *__raddr,
    u_long __program, u_long __version,
    struct timeval __wait_resend,
    int *__sockp, u_int __sendsz,
    u_int __recvsz);

extern struct CLIENT *clntudp_create(struct sockaddr_in *__raddr,
    u_long __program, u_long __version,
    struct timeval __wait_resend,
    int *__sockp);

13.4.42 rpc/pmap_clnt.h

extern u_short pmap_getport(struct sockaddr_in *__address,
    const u_long __program, const u_long __version,
    u_int __protocol);
extern bool_t pmap_set(const u_long __program, const u_long __vers,
    int __protocol, u_short __port);
extern bool_t pmap_unset(u_long __program, u_long __vers);

13.4.43 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
};

#define ar_results ru.AR_results
#define ar_vers ru.AR_versions

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;
        }
    }
};
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;  /* why authentication did not work */
   } ru;
};

#define rj_vers ru.RJ_versions
#define rj_why ru.RJ_why

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;       /* must be equal to two */
   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; /* protocol specific - provided by client */
};

#define rm_call ru.RM_cmb
#define rm_reply ru.RM_rmb
#define acpted_rply ru.RM_rmb.ru.RP_ar
#define rjcted_rply ru.RM_rmb.ru.RP_dr

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 * __xdrs, struct rpc_msg *__cmsg);
extern bool_t xdr_callmsg(XDR * __xdrs, struct rpc_msg *__cmsg);
extern bool_t xdr_rejected_reply(XDR *, struct rejected_reply *);
extern bool_t xdr_replymsg(XDR * __xdrs, struct rpc_msg *__rmsg);

13.4.44 rpc/svc.h

#define svc_getcaller(x)        (&(x)->xp_raddr)
ISO/IEC 23360 Part 1:2008(E)

```c
#define svc_destroy(xprt)       (*(xprt)->xp_ops->xp_destroy)
#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);
};

typedef void (*svc_getreqset(fd_set *__readfds);
extern bool_t svc_register(SVCXPRT * __xprt, rpcprog_t __prog, rpcvers_t __vers, __dispatch_fn_t __dispatch, rpcprot_t __protocol);

extern void svc_run(void);
extern bool_t svc_sendreply(SVCXPRT * xprt, xdrproc_t __xdr_results, caddr_t __xdr_location);
extern void svcerr_auth(SVCXPRT * __xprt, enum auth_stat __why);
extern void svcerr_decode(SVCXPRT * __xprt);
extern void svcerr_noproc(SVCXPRT * __xprt);
```

© 2008 Linux Foundation
extern void svcerr_noprog(SVCXPRT * __xprt);
extern void svcerr_progvers(SVCXPRT * __xprt, rpcvers_t __low_vers,
                             rpcvers_t __high_vers);
extern void svcerr_systemerr(SVCXPRT * __xprt);
extern void svcerr_weakauth(SVCXPRT * __xprt);
extern SVCXPRT *svccrt_create(void);
extern SVCXPRT *svctcp_create(int __sock, u_int __sendsize,
                               u_int __recvsize);
extern SVCXPRT *svcudp_create(int __sock);

13.4.45 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;

13.4.46 rpc/xdr.h

#define XDR_DESTROY(xdrs)  \
    do { if ((xdrs)->x_ops->x_destroy) (*(xdrs)->x_ops->x_destroy)(xdrs);  \
     } while (0)
#define xdr_destroy(xdrs)  \
    do { if ((xdrs)->x_ops->x_destroy) (*(xdrs)->x_ops->x_destroy)(xdrs);  \
     } while (0)
#define XDR_GETBYTES(xdrs,addr,len)      (*(xdrs)->x_ops->x_getbytes)(xdrs, addr, len)
#define xdr_getbytes(xdrs,addr,len)      (*(xdrs)->x_ops->x_getbytes)(xdrs, addr, len)
#define XDR_GETINT32(xdrs,int32p)        (*(xdrs)->x_ops->x_getint32)(xdrs, int32p)
#define xdr_getint32(xdrs,int32p)        (*(xdrs)->x_ops->x_getint32)(xdrs, int32p)
#define XDR_GETLONG(xdrs,longp)          (*(xdrs)->x_ops->x_getlong)(xdrs, longp)
#define xdr_getlong(xdrs,longp)          (*(xdrs)->x_ops->x_getlong)(xdrs, longp)
#define XDR_GETPOS(xdrs)                  (*(xdrs)->x_ops->x_getpostn)(xdrs)
#define xdr_getpos(xdrs)                  (*(xdrs)->x_ops->x_getpostn)(xdrs)
#define XDR_INLINE(xdrs,len)             (*(xdrs)->x_ops->x_inline)(xdrs, len)
#define xdr_inline(xdrs,len)             (*(xdrs)->x_ops->x_inline)(xdrs, len)
#define XDR_PUTBYTES(xdrs,addr,len)      (*(xdrs)->x_ops->x_putbytes)(xdrs, addr, len)
#define xdr_putbytes(xdrs,addr,len)      (*(xdrs)->x_ops->x_putbytes)(xdrs, addr, len)
#define XDR_PUTINT32(xdrs,int32p)        (*(xdrs)->x_ops->x_putint32)(xdrs, int32p)
#define xdr_putint32(xdrs,int32p)        (*(xdrs)->x_ops->x_putint32)(xdrs, int32p)
#define XDR_PUTLONG(xdrs,longp)          (*(xdrs)->x_ops->x_putlong)(xdrs, longp)
#define xdr_putlong(xdrs,longp)          (*(xdrs)->x_ops->x_putlong)(xdrs, longp)
#define XDR_SETPOS(xdrs,pos)             (*(xdrs)->x_ops->x_setpostn)(xdrs)
#define xdr_setpos(xdrs, pos) (*(xdrs)->x_ops->x_setpostn)
(xdrs, pos)

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

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_putbytes) (XDR * __xdrs, char * __addr, u_int __len);
  u_int(*x_getpostn) (XDR * __xdrs);
  bool_t(*x_setpostn) (XDR * __xdrs, u_int __pos);
  int32_t (*x_inline) (XDR * __xdrs, int __len);
  void (*x_destroy) (XDR * __xdrs);
  bool_t(*x_getint32) (XDR * __xdrs, int32_t * __ip);
  bool_t(*x_putint32) (XDR * __xdrs, int32_t * __ip);
};
typedef bool_t(*xdrproc_t) (XDR *, void *, ...);

struct xdr_discrim {
  int value;
  xdrproc_t proc;
};

extern bool_t xdr_array(XDR * _xdrs, caddr_t * __addrp, u_int * __sizep,
                        u_int __maxsize, u_int __elsize,
                        xdrproc_t __elproc);
extern bool_t xdr_bool(XDR * __xdrs, bool_t * __bp);
extern bool_t xdr_bytes(XDR * __xdrs, char **__cpp, u_int * __sizep,
                        u_int __maxsize);
extern bool_t xdr_char(XDR * __xdrs, char * __cp);
extern bool_t xdr_double(XDR * __xdrs, double * __dp);
extern bool_t xdr_enum(XDR * __xdrs, enum_t * __ep);
extern bool_t xdr_float(XDR * __xdrs, float * __fp);
extern void xdr_free(xdrproc_t __proc, char * __objp);
extern bool_t xdr_int(XDR * __xdrs, int * __ip);
extern bool_t xdr_long(XDR * __xdrs, long int * __lp);
extern bool_t xdr_opaque(XDR * __xdrs, caddr_t __cp, u_int __cnt);
extern bool_t xdr_pointer(XDR * __xdrs, char **__objpp, u_int __obj_size,
                          xdrproc_t __xdr_obj);
extern bool_t xdr_reference(XDR * __xdrs, caddr_t * __xpp, u_int __size,
                            xdrproc_t __proc);
extern bool_t xdr_short(XDR * __xdrs, short * __sp);
extern bool_t xdr_string(XDR * __xdrs, char **__cpp, u_int __maxsize);
extern bool_t xdr_u_char(XDR * __xdrs, u_char * __cp);
extern bool_t xdr_u_int(XDR * __xdrs, u_int * __up);
extern bool_t xdr_u_long(XDR * __xdrs, u_long * __ulp);
extern bool_t xdr_u_short(XDR * __xdrs, u_short * __usp);
extern bool_t xdr_union(XDR * __xdrs, enum_t * __dscmp, char * __unp,
    const struct xdr_discrim * __choices, xdrproc_t dfault);
extern bool_t xdr_vector(XDR * __xdrs, char * __basep, u_int __nelem,
    u_int __elemsize, xdrproc_t __xdr_elem);
extern bool_t xdr_void(void);
extern bool_t xdr_wrapstring(XDR * __xdrs, char ** __cpp);
extern void xdrmem_create(XDR * __xdrs, caddr_t __addr, u_int __size,
    enum xdr_op __xop);
extern void xdrrec_create(XDR * __xdrs, u_int __sendsize, u_int __recvsize,
    caddr_t __tcp_handle, int (* __readit)(char *,
    char *,
    int),
    int (* __writeit)(char *, char *,
    int));
extern bool_t xdrrec_endofrecord(XDR * __xdrs, bool_t __sendnow);
extern bool_t xdrrec_eof(XDR * __xdrs);
extern bool_t xdrrec_skiprecord(XDR * __xdrs);
extern void xdrstdio_create(XDR * __xdrs, FILE * __file, enum xdr_op __xop);

13.4.47 sched.h

#define __CPU_EL T(cpu)   ((cpu) / __NCPUBITS)
#define __CPU_M A S K(cpu)  ((__cpu_mask) 1 << ((cpu) % __NCPUBITS))
#define __NCPUBITS      (8 * sizeof (__cpu_mask))
#define SCHED_OTHER     0
#define SCHED_FIFO      1
#define __CPU_SETSIZE   1024
#define SCHED_RR        2
#define CPU_ALLOC(count)        __CPU_ALLOC (count)
#define CPU_ALLOC_SIZE(count)   __CPU_ALLOC_SIZE (count)
#define CPU_COUNT(cpusetp)       __CPU_COUNT_S(sizeof (cpu_set_t), cpusetp)
#define CPU_FREE(cpusetp)        __CPU_FREE (cpusetp)
#define CPU_SETSIZE     __CPU_SETSIZE
#define CPU_ZERO(cpusetp)       __CPU_ZERO_S (sizeof (cpu_set_t), cpusetp)

struct sched_param {
    int sched_priority;
};
typedef unsigned long int __cpu_mask;
typedef struct {
    __cpu_mask __bits[__CPU_SETSIZE / __NCPUBITS];
} cpu_set_t;
extern int sched_get_priority_max(int __algorithm);
extern int sched_get_priority_min(int __algorithm);
extern int sched_getaffinity(pid_t __pid, size_t __cpusetsize,
    cpu_set_t * __cpuset);
extern int sched_getparam(pid_t __pid, struct sched_param * __param);
extern int sched_setscheduler(pid_t __pid);
extern int sched_rr_get_interval(pid_t __pid, struct timespec
extern int sched_setaffinity(pid_t __pid, size_t __cpusetsize, const cpu_set_t *__cpuset);
extern int sched_setparam(pid_t __pid, const struct sched_param *__param);
extern int sched_setscheduler(pid_t __pid, int __policy, const struct sched_param *__param);
extern int sched_yield(void);

13.4.48 search.h

typedef struct entry {
    char *key;
    void *data;
} ENTRY;
typedef enum {
    FIND,
    ENTER
} ACTION;
struct _ENTRY;
typedef enum {
    preorder,
    postorder,
    endorder,
    leaf
} VISIT;
struct hsearch_data {
    struct _ENTRY *table;
    unsigned int size;
    unsigned int filled;
};
typedef void (*__action_fn_t) (const void *__nodep, VISIT __value,
                                int __level);
extern int hcreate(size_t __nel);
extern int hcreate_r(size_t __nel, struct hsearch_data *__htab);
extern void hdestroy(void);
extern void hdestroy_r(struct hsearch_data *__htab);
extern ENTRY *hsearch(ENTRY __item, ACTION __action);
extern int hsearch_r(ENTRY __item, ACTION __action, ENTRY *__retval,
                     struct hsearch_data *__htab);
extern void insque(void *__elem, void *__prev);
extern void *lfind(const void *__key, const void *__base, size_t *__nmemb,
                   size_t __size, __compar_fn_t __compar);
extern void *lsearch(const void *__key, void *__base, size_t *__nmemb,
                     size_t __size, __compar_fn_t __compar);
extern void *remque(void *__elem);
extern void *tdelete(const void *__key, void **__rootp,
                    __compar_fn_t __compar);
extern void *tfind(const void *__key, void **__rootp,
                   __compar_fn_t __compar);
extern void *tsearch(const void *__key, void **__rootp,
                     __compar_fn_t __compar);
extern void *twalk(const void *__root, __action_fn_t __action);

13.4.49 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 __env, int __savemask);
extern void _longjmp(jmp_buf __env, int __val);
extern int _setjmp(jmp_buf __env);
extern void longjmp(jmp_buf __env, int __val);
extern void siglongjmp(sigjmp_buf __env, int __val);

13.4.50 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 NSIG  65
#define SIG_BLOCK    0    /* Block signals. */
#define SIG_UNBLOCK  1    /* Unblock signals. */
#define SIG_SETMASK  2    /* Set the set of blocked
                              signals. */

typedef int sig_atomic_t;

typedef void (*sighandler_t) (int);
#define SIG_HOLD    ((sighandler_t) 2)    /* Request that
                                          signal be held. */
#define SIG_DFL     ((sighandler_t)0)     /* Request for default
                                          signal handling. */
#define SIG_IGN     ((sighandler_t)1)     /* Request that signal be
                                          ignored. */
#define SIG_ERR     ((sighandler_t)-1)    /* Return value from
                                          signal() in case of error. */
#define SIGHUP     1         /* Hangup. */
#define SIGINT     2         /* Terminal interrupt signal. */
#define SIGQUIT    3         /* Terminal quit signal. */
#define SIGILL     4         /* Illegal instruction. */
#define SIGTRAP    5         /* Trace/breakpoint trap. */
#define SIGABRT    6         /* Process abort signal. */
#define SIGIO      6         /* IOT trap */
#define SIGBUS     7         /* Access to an undefined portion
                           of a memory object. */
#define SIGFPE     8         /* Erroneous arithmetic operation. */
#define SIGKILL    9         /* Kill (cannot be caught or
                           ignored). */
#define SIGUSR1   10         /* User-defined signal 1. */
#define SIGSEGV    11        /* Invalid memory reference. */
#define SIGUSR2   12         /* User-defined signal 2. */
#define SIGPIPE   13         /* Write on a pipe with no one
                            to read it. */
#define SIGALRM   14         /* Alarm clock. */
#define SIGTERM   15         /* Termination signal. */
#define SIGSTKFLT 16         /* Stack fault. */
#define SIGCHLD   17         /* Child process terminated,
                           stopped, or continued. */
#define SIGCLD SIGCHLD /* Same as SIGCHLD */
#define SIGCONT 18 /* Continue executing, if stopped. */
#define SIGSTOP 19 /* Stop executing (cannot be caught or ignored). */
#define SIGTSTP 20 /* Terminal stop signal. */
#define SIGTTIN 21 /* Background process attempting read. */
#define SIGTTOU 22 /* Background process attempting write. */
#define SIGURG 23 /* High bandwidth data is available at a socket. */
#define SIGXCPU 24 /* CPU time limit exceeded. */
#define SIGXFSZ 25 /* File size limit exceeded. */
#define SIGVTALRM 26 /* Virtual timer expired. */
#define SIGPROF 27 /* Profiling timer expired. */
#define SIGWINCH 28 /* Window size change. */
#define SIGIO 29 /* I/O now possible. */
#define SIGPOLL SIGIO /* Pollable event. */
#define SIGPWR 30 /* Power failure restart */
#define SIGSYS 31 /* Bad system call. */
#define SIGUNUSED 31
#define SV_ONSTACK (1<<0) /* Take the signal on the signal stack. */
#define SV_INTERRUPT (1<<1) /* Do not restart system calls. */
#define SV_RESETHAND (1<<2) /* Reset handler to SIG_DFL on receipt. */

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

#define SIGEV_SIGNAL 0 /* Notify via signal. */
#define SIGEV_NONE 1 /* Other notification: meaningless. */
#define SIGEV_THREAD 2 /* Deliver via thread creation. */
#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
```c
#define si_fd   _sifields._sigpoll._fd
#define si_timer1       _sifields._timer._timer1
#define si_timer2       _sifields._timer._timer2

typedef struct siginfo {
    int si_signo;               /* Signal number. */
    int si_errno;               /* Error number. */
    int si_code;                /* Signal 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      /* Sent by sigqueue. */
#define SI_TIMER        -2      /* Sent by timer expiration. */
#define SI_MESGQ        -3      /* Sent by real time mesq state change. */
#define SI_ASYNCIO      -4      /* Sent by AIO completion. */
#define SI_SIGIO        -5      /* Sent by queued SIGIO. */
#define SI_TKILL       -6       /* Sent by tkill. */
#define SI_ASYNCNL      -60     /* Sent by asynch name lookup completion. */
#define SI_USER        0       /* Sent by kill, sigsend, raise. */
#define SI_KERNEL      0x80     /* Sent by kernel. */
#define ILL_ILLOPC      1       /* Illegal opcode. */
#define ILL_ILLOPN      2       /* Illegal operand. */
#define ILL_ILLADR      3       /* Illegal addressing mode. */
#define ILL_ILLTRP      4       /* Illegal trap. */
#define ILL_PRVOPC      5       /* Privileged opcode. */
#define ILL_PRVREG      6       /* Privileged register. */
#define ILL_COPROC      7       /* Coprocessor error. */
#define ILL_BADSTK      8       /* Internal stack error. */
#define FPE_INTDIV      1       /* Integer divide by zero. */
#define FPE_INTOVF      2       /* Integer overflow. */
#define FPE_FLTDIV      3       /* Floating-point divide by zero. */
```
/* Floating-point overflow. */
#define FPE_FLTOVF 4
/* Floating-point underflow. */
#define FPE_FLTUND 5
/* Floating-point inexact result. */
#define FPE_FLTRES 6
/* Invalid floating-point operation. */
#define FPE_FLTINV 7
/* Subscript out of range. */
#define FPE_FLTSUB 8

/* Address not mapped to object. */
#define SEGV_MAPERR 1
/* Invalid permissions for mapped object. */
#define SEGV_ACCERR 2
/* Invalid address alignment. */
#define BUS_ADRALN 1
/* Nonexistent physical address. */
#define BUS_ADRERR 2
/* Object-specific hardware error. */
#define BUS_OBJERR 3

/* Process breakpoint. */
#define TRAP_BRKPT 1
/* Process trace trap. */
#define TRAP_TRACE 2
/* Child has exited. */
#define CLD_EXITED 1
/* Child has terminated abnormally and did not create a core file. */
#define CLD_KILLED 2
/* Child has terminated abnormally and created a core file. */
#define CLD_DUMPED 3
/* Traced child has trapped. */
#define CLD_TRAPPED 4
/* Child has stopped. */
#define CLD_STOPPED 5
/* Stopped child has continued. */
#define CLD_CONTINUED 6

/* Data input available. */
#define POLL_IN 1
/* Output buffers available. */
#define POLL_OUT 2
/* Input message available. */
#define POLL_MSG 3
/* I/O error. */
#define POLL_ERR 4
/* High priority input available. */
#define POLL_PRI 5
/* Device disconnected. */
#define POLL_HUP 6

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

/* Use signal stack by using `sa_restorer`. */
#define SA_INTERRUPT 0x20000000
#define sa_handler __sigaction_handler.__sa_handler
#define sa_sigaction __sigaction_handler.__sa_sigaction
#define SA_ONSTACK 0x08000000 /* Use signal stack by entry to handler. */
#define SA_RESETHAND 0x80000000 /* Don't send SIGCHLD when children stop. */
#define SA_SIGINFO 0x00000004 /* Invoke signal-catching function with three arguments instead of one. */
#define SA_ONCLDWAIT 0x00000001 /* Don't send SIGCHLD when children stop. */
#define SA_NODESIGNO 0x00000004 /* Don't automatically block the signal when its handler is being executed. */
#define SA_ONOCLDWAIT 0x00000002 /* Don't create zombie on child death. */
#define SA_NOMASK SA_NODEFER
#define SA_ONESHOT SA_RESETHAND

typedef struct sigaltstack {
    void *ss_sp;
} sigaltstack;
```c
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 __sig, sighandler_t __handler);
extern int __xpg_sigpause(int);
extern char *const _sys_siglist[];
extern sighandler_t bsd_signal(int __sig, sighandler_t __handler);
extern int kill(pid_t __pid, int __sig);
extern int killpg(pid_t __pgid, int __sig);
extern void psignal(int __sig, const char *__s);
extern int pthread_kill(pthread_t, int);
extern int pthread_sigmask(int, const sigset_t *, sigset_t *);
extern int __sysv_signal(int, sighandler_t);
extern int sigaction(int __sig, const struct sigaction *__act, struct sigaction *__oact);
extern int sigaddset(sigset_t * __set, int __signo);
extern int sigaltstack(const struct sigaltstack *__ss, struct sigaltstack *__oss);
extern int sigandset(sigset_t * __set, const sigset_t * __left, const sigset_t * __right);
extern int sigemptyset(sigset_t * __set);
extern int sigfillset(sigset_t * __set);
extern int sighold(int __sig);
extern int sigignore(int __sig);
extern int siginterrupt(int __sig, int __interrupt);
extern int sigisemptyset(const sigset_t * __set);
extern int sigismember(const sigset_t * __set, int __signo);
extern sighandler_t signal(int __sig, sighandler_t __disp);
extern int sigsuspend(const sigset_t * __set);
extern int sigtimedwait(const sigset_t * __set, siginfo_t * __info, const struct timespec *__timeout);
extern int sigwait(const sigset_t * __set, int *__sig);
extern int sigwaitinfo(const sigset_t * __set, siginfo_t * __info);

13.4.51 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
```
```c
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;
    struct sched_param __sp;
    int __policy;
    int __pad[16];
} posix_spawnattr_t;

extern int posix_spawn(pid_t * __pid, const char *__path,
                        const posix_spawn_file_actions_t * __file_actions,
                        const posix_spawnattr_t * __attrp,
                        char *const argv[], char *const envp[]);

extern int posix_spawn_file_actions_addclose(posix_spawn_file_actions_t * __file_actions, int __fd);
extern int posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t * __file_actions, int __fd, int __newfd);
extern int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t * __file_actions, int __fd,
                                            const char *__path, int __oflag, mode_t __mode);

extern int posix_spawn_file_actions_destroy(posix_spawn_file_actions_t * __file_actions);

extern int posix_spawn_file_actions_init(posix_spawn_file_actions_t * __file_actions);

extern int posix_spawnattr_destroy(posix_spawnattr_t * __attr);
extern int posix_spawnattr_getflags(const posix_spawnattr_t * __attr,
                                     short int *__flags);
extern int posix_spawnattr_getpgroup(const posix_spawnattr_t * __attr,
                                      pid_t * __pgroup);
extern int posix_spawnattr_getschedparam(const posix_spawnattr_t * __attr,
                                          struct sched_param * __schedparam);
extern int posix_spawnattr_getschedpolicy(const posix_spawnattr_t * __attr,
                                         int * __schedpolicy);
extern int posix_spawnattr_getsigdefault(const posix_spawnattr_t * __attr,
                                         sigset_t * __sigdefault);
extern int posix_spawnattr_getsigmask(const posix_spawnattr_t * __attr,
                                       sigset_t * __sigmask);
extern int posix_spawnattr_init(posix_spawnattr_t * __attr);
```
extern int posix_spawnp(pid_t *__pid, const char *__file, const posix_spawn_file_actions_t *__file_actions, const posix_spawnattr_t *__attrp, char *const argv[], char *const envp[]);

13.4.52 stddef.h

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

13.4.53 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 # U
#define INT8_MIN        (-128)
#define INT_FAST8_MIN   (-128)
#define INT_LEAST8_MIN  (-128)
#define INT32_MIN       (-2147483647)
#define INT_FAST32_MIN  (-2147483647)
#define INT_LEAST32_MIN (-2147483647)
#define INT16_MIN       (-32767)
#define INT_FAST16_MIN  (-32767)
#define INT_LEAST16_MIN (-32767)
#define INT64_MIN       (-__INT64_C(9223372036854775807))
#define INTMAX_MIN      (-__INT64_C(9223372036854775807))
#define INT_FAST64_MIN  (-__INT64_C(9223372036854775807))
#define INT_LEAST64_MIN (-__INT64_C(9223372036854775807))
#define WINT_MIN        (0u)
#define INT8_MAX        (127)
#define INT_FAST8_MAX   (127)
#define INT_LEAST8_MAX  (127)
#define INT32_MAX       (2147483647)
#define INT_FAST32_MAX  (2147483647)
#define INT16_MAX       (32767)
#define INT_FAST16_MAX  (32767)
#define INT8_MAX        (255)
#define INT_FAST8_MAX   (255)
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;

13.4.54 stdio.h

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

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

typedef struct _IO_FILE FILE;

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

extern char *__fgets_chk(char *, size_t, int, FILE *);
extern char *__fgets_unlocked_chk(char *, size_t, int, FILE *);
extern size_t __fpending(FILE *);
extern int __printf_chk(int, const char *, ...);
extern int __snprintf_chk(char *, size_t, int, size_t, const char *...);
extern int __sprintf_chk(char *, int, size_t, const char *, ...);
extern int __vprintf_chk(int, const char *, va_list);
extern int __vsnprintf_chk(char *, size_t, int, size_t, const char *,
                           va_list);
extern int __vsprintf_chk(char *, int, size_t, const char *, va_list);
extern char *const _sys_errlist[];
extern int asprintf(char **__ptr, const char *__fmt, ...);
extern void clearerr(FILE * __stream);
extern void clearerr_unlocked(FILE * __stream);
extern int dprintf(int __fd, const char *__fmt, ...);
extern int fclose(FILE * __stream);
extern FILE *fdopen(int __fd, const char *__modes);
extern int feof(FILE * __stream);
extern int feof_unlocked(FILE * __stream);
extern int ferror(FILE * __stream);
extern int ferror_unlocked(FILE * __stream);
extern int fflush(FILE * __stream);
extern int fflush_unlocked(FILE * __stream);
extern int fgetc(FILE * __stream);
extern int fgetc_unlocked(FILE * __stream);
extern int fgetpos(FILE * __stream, fpos_t * __pos);
extern int fgetpos64(FILE * __stream, fpos64_t * __pos);
extern char *fgets(char *__s, int __n, FILE * __stream);
extern char *fgets_unlocked(char *__s, int __n, FILE * __stream);
extern int fileno(FILE * __stream);
extern int fileno_unlocked(FILE * __stream);
extern void flockfile(FILE * __stream);
extern FILE *fmemopen(void  *__s, size_t  __len, const  char  *__modes);
extern FILE *fopen(const char *__filename, const char *__modes);
extern FILE *fopen64(const char *__filename, const char *__modes);
extern int fprintf(FILE * __stream, const char *__format, ...);
extern int fputc(int __c, FILE * __stream);
extern int fputc_unlocked(int __c, FILE * __stream);
extern int fputs(const char *__s, FILE * __stream);
extern int fputs_unlocked(const char *__s, FILE * __stream);
extern size_t fread(void *__ptr, size_t __size, size_t __n,
                     FILE * __stream);
extern size_t fread_unlocked(void *__ptr, size_t __size, size_t __n,
                              FILE * __stream);
extern FILE *freopen(const char *__filename, const char *__modes, FILE * __stream);
extern FILE *freopen64(const char *__filename, const char *__modes,
                       FILE * __stream);
extern int fscanf(FILE * __stream, const char *__format, ...);
extern int fseek(FILE * __stream, long int __off, int __whence);
extern int fseeko(FILE * __stream, off_t __off, int __whence);
extern int fseeko64(FILE * __stream, loff_t __off, int __whence);
extern int fsetpos(FILE * __stream, const fpos_t * __pos);
extern int fsetpos64(FILE * __stream, const fpos64_t * __pos);
extern long int ftello(FILE * __stream);
extern off_t ftello64(FILE * __stream);
extern int ftrylockfile(FILE * __stream);
extern void funlockfile(FILE * __stream);
extern size_t fwrite(const void *__ptr, size_t __size, size_t __n,
                     FILE * __stream);
extern size_t fwrite_unlocked(const void *__ptr, size_t __size, size_t __n,
FILE *__stream);
extern int getc(FILE *__stream);
extern int getc_unlocked(FILE *__stream);
extern int getchar(void);
extern int getchar_unlocked(void);
extern ssize_t getdelim(char **__lineptr, size_t *__n, int __delimiter,
    FILE *__stream);
extern ssize_t getline(char **__lineptr, size_t *__n, FILE *__stream);
extern int getw(FILE *__stream);
extern FILE *open_memstream(char **__bufloc, size_t *__sizeloc);
extern int pclose(FILE *__stream);
extern void perror(const char *__s);
extern FILE *popen(const char *__command, const char *__modes);
extern int printf(const char *__format, ...);
extern int putc(int __c, FILE *__stream);
extern int putc_unlocked(int __c, FILE *__stream);
extern int putchar(int __c);
extern int putchar_unlocked(int __c);
extern int putw(int __w, FILE *__stream);
extern int remove(const char *__filename);
extern int renameat(int __oldfd, const char *__old, int __newfd,
    const char *__new);
extern void rewind(FILE *__stream);
extern int scanf(const char *__format, ...);
extern void setbuf(FILE *__stream, char *__buf);
extern void setbuffer(FILE *__stream, char *__buf, size_t __size);
extern int setvbuf(FILE *__stream, char *__buf, int __modes,
    size_t __n);
extern int snprintf(char *__s, size_t __maxlen, const char *__format, ...);
extern int sprintf(char *__s, const char *__format, ...);
extern int sscanf(const char *__s, const char *__format, ...);
extern FILE *stderr;
extern FILE *stdin;
extern FILE *stdout;
extern char *tempnam(const char *__dir, const char *__pfx);
extern FILE *tmpfile(void);
extern FILE *tmpfile64(void);
extern char *tmpnam(char *__s);
extern int ungetc(int __c, FILE *__stream);
extern int vasprintf(char **__ptr, const char *__f, va_list __arg);
extern int vdprintf(int __fd, const char *__fmt, va_list __arg);
extern int vfprintf(FILE *__s, const char *__format, va_list __arg);
extern int vfscanf(FILE *__s, const char *__format, va_list __arg);
extern int vprintf(const char *__format, va_list __arg);
extern int vscanf(const char *__format, va_list __arg);
extern int vsnprintf(char *__s, size_t __maxlen, const char *__format, va_list __arg);
extern int vsprintf(char *__s, const char *__format, va_list __arg);
extern int vsscanf(const char *__s, const char *__format, va_list __arg);

13.4.55 stdlib.h

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

struct drand48_data {
    unsigned short __x[3];
    unsigned short __old_x[3];
    unsigned short __c;
    unsigned short __init;
    unsigned long long int __a;
};
typedef int (*__compar_fn_t) (const void *, const void *);
struct random_data {
    int32_t *fptr;   /* Front pointer. */
    int32_t *rptr;   /* Rear pointer. */
    int32_t *state; /* Array of state values. */
    int rand_type;  /* Type of random number generator. */
    int rand_deg;   /* Degree of random number generator. */
    int rand_sep;   /* Distance between front and rear. */
    int32_t *end_ptr; /* Pointer behind state table. */
};
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;
extern void _Exit(int __status);
extern size_t __ctype_get_mb_cur_max(void);
extern size_t __mbstowcs_chk(wchar_t *, const char *, size_t, size_t);
extern char *__realpath_chk(const char *, char *, size_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, int);
extern long int __strto1l_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 size_t __wcstombs_chk(char *, const wchar_t *, size_t, size_t);
extern int __wcrtomb_chk(char *, wchar_t, size_t);
extern long int a64l(const char *__s);
extern void abort(void);
extern int abs(int __x);
extern double atof(const char *__nptr);
extern int atoi(const char *__nptr);
```
extern char *ptsname(int __fd);
extern int putenv(char *__string);
extern void qsort(void *__base, size_t __nmemb, size_t __size,
              const __compar_fn_t __compar);
extern int rand(void);
extern int rand_r(unsigned int *__seed);
extern long int random(void);
extern int random_r(struct random_data *__buf, int32_t *__result);
extern void *realloc(void *__ptr, size_t __size);
extern char *realpath(const char *__name, char *__resolved);
extern unsigned short *seed48(unsigned short __seed16v[3]);
extern int seed48_r(unsigned short __seed16v[3],
              struct drand48_data *__buffer);
extern void *seed48_r(unsigned short __seed16v[3],
              struct drand48_data *__buffer);
extern int setenv(const char *__name, const char *__value, int __replace);
extern char *setstate(char *__statebuf);
extern int setstate_r(char *__statebuf, struct random_data *__buf);
extern void srand(unsigned int __seed);
extern void srand48(long int __seedval);
extern int srand48_r(long int __seedval, struct drand48_data *__buffer);
extern int setstate_r(char *__statebuf, struct random_data *__buf);
extern double strtod(const char *__nptr, char **__endptr);
extern float strtof(const char *__nptr, char **__endptr);
extern long int strtol(const char *__nptr, char **__endptr, int __base);
extern long double strtold(const char *__nptr, char **__endptr);
extern long long int strtoll(const char *__nptr, char **__endptr,
               int __base);
extern void __memcpy_chk(void *, const void *, size_t, size_t);
extern void __memmove_chk(void *, const void *, size_t, size_t);
extern void *__mempcpy(void *__dest, const void *__src, size_t __n);
extern void *__mempcpy_chk(void *, const void *, size_t, size_t);
extern void *__memset_chk(void *, int, size_t, size_t);
extern char *__stpcpy(char *__dest, const char *__src);
extern char *__stpcpy_chk(char *, const char *, size_t);
extern char *__strcat_chk(char *, const char *, size_t);
extern int system(const char *__command);
extern int unlockpt(int __fd);
extern int unsetenv(const char *__name);
extern size_t wcstombs(char *__s, const wchar_t * __pwcs, size_t __n);
extern int wctomb(char *__s, wchar_t __wchar);

13.4.56 string.h

#define __xpg_strerror_r

extern void *__memcpy_chk(void *, const void *, size_t, size_t);
extern void *__memmove_chk(void *, const void *, size_t, size_t);
extern void *__mempcpy(void *, const void *, size_t, size_t);
extern void *__mempcpy_chk(void *, const void *, size_t, size_t);
extern void *__memset_chk(void *, int, size_t, size_t);
extern char *__stpcpy(char *, const char *, size_t);
extern char *__strcat_chk(char *, const char *, size_t);
extern char *__strcpy_chk(char *, const char *, size_t);
extern char *__strncat_chk(char *, const char *, size_t, size_t);
extern char *__strncpy_chk(char *, const char *, size_t, size_t);
extern char *__strtok_r(char *__s, const char *__delim, char **__save_ptr);
extern int __xpg_strerror_r(int, char *, size_t);
extern void *memccpy(void *__dest, const void *__src, int __c, size_t __n);
extern char *stpcpy(char *__dest, const char *__src);
extern char *stpncpy(char *__dest, const char *__src, size_t __n);
extern char *strcasestr(const char *__haystack, const char *__needle);
extern char *strcat(char *__dest, const char *__src);
extern char *strchr(const char *__s, int __c);
extern int strcmp(const char *__s1, const char *__s2);
extern int strcoll(const char *__s1, const char *__s2);
extern char *strcpy(char *__dest, const char *__src);
extern size_t strcspn(const char *__s, const char *__reject);
extern char *strdup(const char *__s);
extern char *strerror(int __errnum);
extern size_t strlen(const char *__s);
extern char *strncat(char *__dest, const char *__src, size_t __n);
extern int strncmp(const char *__s1, const char *__s2, size_t __n);
extern char *strncpy(char *__dest, const char *__src, size_t __n);
extern char *strndup(const char *__string, size_t __n);
extern size_t strnlen(const char *__string, size_t __maxlen);
extern char *strpbrk(const char *__s, const char *__accept);
extern int ffs(int __i);
extern char *index(const char *__s, int __c);
extern char *rindex(const char *__s, int __c);
extern int strcasecmp(const char *__s1, const char *__s2);
extern int strncasecmp(const char *__s1, const char *__s2, size_t __n);
extern int strndup(const char *__string, size_t __n);
extern size_t strnlen(const char *__string, size_t __maxlen);
extern char *strsep(const char **__stringp, const char *__delim);
extern char *strsignal(int __sig);
extern size_t strspn(const char *__s, const char *__accept);
extern char *strstr(const char *__haystack, const char *__needle);
extern char *strtok(char *__s, const char *__delim);
extern char *strtok_r(char *__s, const char *__delim, char **__save_ptr);
extern size_t strxfrm(char *__dest, const char *__src, size_t __n);

13.4.57 strings.h

extern int bcmp(const void *__s1, const void *__s2, size_t __n);
extern void bcopy(const void *__src, void *__dest, size_t __n);
extern void bzero(void *__s, size_t __n);
extern int ffs(int __i);
extern char *index(const char *__s, int __c);
extern char *rindex(const char *__s, int __c);
extern int strcasecmp(const char *__s1, const char *__s2);
extern int strcasecmp(const char *__s1, const char *__s2, size_t __n);

13.4.58 sys/epoll.h

#define EPOLL_CTL_ADD   1   /* Add a file descriptor to the interface. */
#define EPOLL_CTL_DEL   2   /* Remove a file descriptor from the interface. */
#define EPOLL_CTL_MOD   3   /* Change file descriptor epoll_event structure. */
#define EPOLLIN 1
#define EPOLLPRI        2
#define EPOLLOUT        4
#define EPOLLERR        8
#define EPOLLHUP 16
#define EPOLLRDHUP 32
#define EPOLLONESHOT (1 << 30)
#define EPOLLET (1 << 31)

typedef union epoll_data {
    void *ptr;
    int fd;
    uint32_t u32;
    uint64_t u64;
} epoll_data_t;

struct epoll_event {
    uint32_t events;
    epoll_data_t data;
};

extern int epoll_create(int __size);
extern int epoll_ctl(int __epfd, int __op, int __fd, struct epoll_event *__event);
extern int epoll_wait(int __epfd, struct epoll_event *__events, int __maxevents, int __timeout);

13.4.59 sys/file.h

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

extern int flock(int __fd, int __operation);

13.4.60 sys/inotify.h

#define IN_ACCESS       0x00000001
#define IN_MODIFY       0x00000002
#define IN_ATTRIB       0x00000004
#define IN_CLOSE_WRITE 0x00000008
#define IN_CLOSE_NOWRITE 0x00000000
#define IN_OPEN 0x00000020
#define IN_MOVED_FROM 0x00000040
#define IN_MOVED_TO 0x00000080
#define IN_CREATE 0x00000100
#define IN_DELETE 0x00000200
#define IN_DELETE_SELF 0x00000400
#define IN_MOVE_SELF 0x00000800
#define IN_UNMOUNT 0x00002000
#define IN_Q_OVERFLOW 0x00004000
#define IN_IGNORED 0x00008000
#define IN_ISDIR 0x40000000
ISO/IEC 23360 Part 1:2008(E)

#define IN_ONESHOT 0x80000000
#define IN_CLOSE (IN_CLOSE_WRITE | IN_CLOSE_NOWRITE)
#define IN_MOVE (IN_MOVED_FROM | IN_MOVED_TO)
#define IN_ALL_EVENTS \  
  (IN_ACCESS | IN_MODIFY | IN_ATTRIB | IN_CLOSE_WRITE | \  
  IN_CLOSE_NOWRITE | IN_OPEN | IN_MOVED_FROM | IN_MOVED_TO | \  
  IN_CREATE | \  
  IN_DELETE | IN_DELETE_SELF | IN_MOVE_SELF)

struct inotify_event {
    int wd;
    uint32_t mask;
    uint32_t cookie;
    uint32_t len;
    char name[];
};
extern int inotify_add_watch(int __fd, const char *__name, uint32_t __mask);
extern int inotify_init(void);
extern int inotify_rm_watch(int __fd, uint32_t __wd);

13.4.61 sys/ioctl.h

struct winsize {
    unsigned short ws_row; /* Rows, in characters. */
    unsigned short ws_col; /* Columns, in characters. */
    unsigned short ws_xpixel; /* Horizontal pixels. */
    unsigned short ws_ypixel; /* Vertical pixels. */
};
extern int ioctl(int __fd, unsigned long int __request, ...);

13.4.62 sys/ipc.h

#define IPC_PRIVATE 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
#define ftok(const char *__pathname, int __proj_id);

13.4.63 sys/mman.h

#define MAP_FAILED (void*)-1
#define MAP_SHARED 0x01
#define MAP_PRIVATE 0x02
#define PROT_READ 0x1
#define PROT_WRITE 0x2
#define PROT_EXEC 0x4
#define MREMAP_MAYMOVE 1
#define MS_ASYNC 1
#define MS_INVALIDATE 2
#define MS_ASYNC 2
#define POSIX_MADV_WILLNEED 3
#define MS_SYNC 4
#define POSIX_MADV_DONTNEED 4
#define MAP_ANON MAP_ANONYMOUS

extern int mlock(const void *__addr, size_t __len);
extern int mlockall(int __flags);
extern void *mmap(void *__addr, size_t __len, int __prot, int __flags,
               int __fd, off_t __offset);
extern void *mmap64(void *__addr, size_t __len, int __prot, int __flags,
                    int __fd, off64_t __offset);
extern int mprotect(void *__addr, size_t __len, int __prot);
extern void *mremap(void *__addr, size_t __old_len, size_t __new_len,
                    int __flags, ...);
extern int msync(void *__addr, size_t __len, int __flags);
extern int munlock(const void *__addr, size_t __len);
extern int munlockall(void);
extern int munmap(void *__addr, size_t __len);
extern int posix_madvise(void *__addr, size_t __len, int __advice);
extern int shm_open(const char *__name, int __oflag, mode_t __mode);
extern int shm_unlink(const char *__name);

13.4.64 sys/msg.h

#define MSG_NOERROR 010000
extern int msctl(int __msqid, int __cmd, struct msqid_ds *__buf);
extern int msgget(key_t __key, int __msgflg);
extern ssize_t msgrcv(int __msqid, void __msgp, size_t __msgsz,
                      long __msgtyp, int __msgflg);
extern int msgsnd(int __msqid, const void __msgp, size_t __msgsz,
                  int __msgflg);

13.4.65 sys/param.h

#define NOFILE 256
#define MAXPATHLEN 4096

13.4.66 sys/poll.h

#define POLLIN 0x0001 /* There is data to read */
#define POLLPRI 0x0002 /* There is urgent data to read */
#define POLLOUT 0x0004 /* Writing now will not block */
#define POLLRERR 0x0008 /* Error condition */
#define POLLHUP 0x0010 /* Hung up */
#define POLLINVAL 0x0020 /* Invalid request: fd not open */
#define POLLRDNORM 0x0040 /* Normal data may be read */
#define POLLRDNDINT 0x0080 /* Priority data may be read */
#define POLLWRNORM 0x0100 /* Writing now will not block */
#define POLLWRNDINT 0x0200 /* Priority data may be written */
struct pollfd {
    int fd; /* File descriptor to poll. */
    short events; /* Types of events poller cares about. */
    short revents; /* Types of events that actually occurred. */
};

typedef unsigned long int nfds_t;

13.4.67 sys/resource.h

#define RUSAGE_CHILDREN (-1)
#define RLIM_INFINITY (~0UL)
#define RLIM_SAVED_CUR -1
#define RLIM_SAVED_MAX -1
#define RLIMIT_CPU 0
#define RUSAGE_SELF 0
#define RLIMITFSIZE 1
#define RLIMITLOCKS 10
#define RLIMNLIMITS 11
#define RLIMITSTACK 3
#define RLIMITCORE 4
#define RLIMITRSS 5
#define RLIMITNPAGES 6
#define RLIMITNOFILE 7
#define RLIMITMEMLOCK 8
#define RLIMITAS 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; /* The current (soft) limit. */
    rlim_t rlim_max; /* The hard limit. */
};

struct rlimit64 {
    rlim64_t rlim_cur; /* The current (soft) limit. */
    rlim64_t rlim_max; /* The hard limit. */
};

struct rusage {
    struct timeval ru_utime; /* Total amount of user time used. */
    struct timeval ru_stime; /* Total amount of system time used. */
    long int ru_maxrss; /* Maximum resident set size (in kilobytes). */
    long int ru_ixrss; /* Amount of sharing of text segment memory with other p */
    long int ru_idrss; /* Amount of data segment memory used (kilobyte-seconds). */
    long int ru_isrss; /* Amount of stack memory used (kilobyte-seconds). */
    long int ru_minflt; /* Number of soft page faults (i.e. those serviced by reclaimin */
    long int ru_majflt; /* Number of hard page faults (i.e. those that required I/O). */
    long int ru_nswap; /* Number of times a process was swapped out of physical memory */
    long int ru_inblock; /* Number of input operations via the file system. Note: This */
    long int ru_outblock; /* Number of output operations */
};
via the file system. */
    long int ru_msgsnd;    /* Number of IPC messages sent. */
    long int ru_msgrcv;    /* Number of IPC messages received. */
    long int ru_nsignals;  /* Number of signals delivered. */
    long int ru_nvcsw;     /* Number of voluntary context switches, i.e. because the proce */
    long int ru_nivcsw;    /* Number of involuntary context switches, i.e. a higher priori */
};

enum __priority_which {
    PRIO_PROCESS = 0,        /* WHO is a process ID. */
    PRIO_PGRP = 1,            /* WHO is a process group ID. */
    PRIO_USER = 2             /* WHO is a user ID. */
};

#define PRIO_PGRP       PRIO_PGRP
#define PRIO_PROCESS    PRIO_PROCESS
#define PRIO_USER       PRIO_USER

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

13.4.68 sys/select.h

#define NFDBITS (8 * sizeof (long))
extern int pselect(int __nfds, fd_set *__readfds, fd_set *__writefds,
                   fd_set *__exceptfds, const struct timespec *__timeout,
                   const sigset_t *__sigmask);

13.4.69 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 __semid, int __semnum, int __cmd, ...);
extern int semget(key_t __key, int __nsems, int __semflg);
extern int semop(int __semid, struct sembuf *__sops, size_t __nsops);

13.4.70 sys/sendfile.h

extern ssize_t sendfile(int __out_fd, int __in_fd, off_t *__offset,
                         size_t __count);
extern ssize_t sendfile64(int __out_fd, int __in_fd, off64_t *__offset,
                          size_t __count);

13.4.71 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 __shmid, const void *__shmaddr, int __shmflg);
extern int shmctl(int __shmid, int __cmd, struct shmid_ds *__buf);
extern int shmdt(const void *__shmaddr);
extern int shmget(key_t __key, size_t __size, int __shmflg);

13.4.72 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)  (((len)+sizeof(size_t)-1)&(size_t)~(sizeof(size_t)-1))
#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 *)((u_char *)(msg) + CMSG_ALIGN((msg)->cmsg_len)) : \  (struct cmsghdr *)NULL)
#define CMSG_NXTHDR(mhdr,cmsg)  \  (((cmsg) == NULL) ? CMSG_FIRSTHDR(mhdr) : \  (((u_char *)(cmsg) + CMSG_ALIGN((cmsg)->cmsg_len)) > \  (u_char *)((mhdr)->msg_control) + (mhdr)->msg_controllen) ? \  (struct cmsghdr *)(u_char *)((mhdr)->msg_control + (mhdr)->msg_controllen) : \  (struct cmsghdr *)NULL : \  (struct cmsghdr *)(u_char *)((mhdr)->msg_control + CMSG_ALIGN((mhdr)->msg_controllen))))
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_aligntype)))];
};

struct msghdr {
    void *msg_name;
    int msg_namelen;
    struct iovec *msg_iov;
    size_t msg_iovlen;
    void *msg_control;
    size_t msg_controllen;
    unsigned int msg_flags;
};

#define AF_UNSPEC       0
#define AF_UNIX         1
#define AF_INET6        10
#define AF_INET         2
#define PF_INET AF_INET
#define PF_INET6 AF_INET6
#define PF_UNIX AF_UNIX
#define 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_BSDCOMPAT    14
#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 SIOCGIFMTU 0x8917
#define SIOCGIFDSTADDR 0x8919
#define SIOCGIFNETMASK 0x891b
#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_NOSIGNAL 0x4000
#define MSG_EOR 0x80
#define MSG_OOB 1
#define MSG_PEEK 2
#define MSG_DONTROUTE 4
#define MSG_CTRUNC 8

extern ssize_t __recv_chk(int, void *, size_t, size_t, int);
extern ssize_t __recvfrom_chk(int, void *, size_t, size_t, int,
    struct sockaddr *, socklen_t *);
extern int accept(int __fd, struct sockaddr *__addr,
    socklen_t *__addr_len);
extern int bind(int __fd, const struct sockaddr *__addr,
    socklen_t __len);
extern int connect(int __fd, const struct sockaddr *__addr,
    socklen_t __len);
extern int getnameinfo(const struct sockaddr *__sa, socklen_t __salen,
    char *__host, socklen_t __hostlen, char *__serv,
    socklen_t __servlen, unsigned int __flags);
extern int getpeername(int __fd, struct sockaddr *__addr,
    socklen_t *__len);
extern int getsockname(int __fd, struct sockaddr *__addr,
    socklen_t *__len);
extern int getsockopt(int __fd, int __level, int __optname, void *__optval,
    socklen_t *__optlen);
extern int listen(int __fd, int __n);
extern ssize_t recv(int __fd, void *__buf, size_t __n, int __flags);
extern ssize_t recvfrom(int __fd, void *__buf, size_t __n, int __flags,
    struct sockaddr *__addr, socklen_t *__addr_len);
extern ssize_t recvmsg(int __fd, struct msghdr *__message, int __flags);
extern ssize_t send(int __fd, const void *__buf, size_t __n, int __flags);
extern ssize_t sendmsg(int __fd, const struct msghdr *__message,
    int __flags);
extern ssize_t sendto(int __fd, const void *__buf, size_t __n, int __flags,
    const struct sockaddr *__addr, socklen_t __len);
13 Base Libraries

ISO/IEC 23360 Part 1:2008(E)

extern int setsockopt(int __fd, int __level, int __optname,
                        const void *__optval, socklen_t __optlen);
extern int shutdown(int __fd, int __how);
extern int sockatmark(int __fd);
extern int socket(int __domain, int __type, int __protocol);
extern int socketpair(int __domain, int __type, int __protocol,
                        int __fds[2]);

13.4.73 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_ISSOCK(m)     (((m)&S_IFMT)==S_IFSOCK)
#define S_TYPEISMQ(buf) ((buf)->st_mode - (buf)->st_mode)
#define S_TYPEISEM(buf)        ((buf)->st_mode - (buf)->st_mode)
#define S_TYPEISSHM(buf)        ((buf)->st_mode - (buf)->st_mode)
#define S_IRWXU (S_IREAD|S_IWRITE|S_IEXEC)
#define S_IROTH (S_IRGRP>>3)
#define S_IRGRP (S_IRUSR>>3)
#define S_IRWXO (S_IRWXG>>3)
#define S_IRWXG (S_IRWXU>>3)
#define S_IWOTH (S_IWGRP>>3)
#define S_IWGRP (S_IWUSR>>3)
#define S_IXOTH (S_IXGRP>>3)
#define S_IXGRPX (S_IXUSR>>3)
#define S_ISVTX 01000
#define S_IWUSR 0x0040
#define S_IWUSR 0x0080
#define S_ISGID 0x0400
#define S_ISUID 0x0800
#define S_IFIFO 0x1000
#define S_IFDIR 0x4000
#define S_IFBLK 0x6000
#define S_IFSOCK 0xc000
#define S_IFMT 0xf000
#define st_atime        st_atim.tv_sec
#define st_ctime        st_ctim.tv_sec
#define st_mtime        st_mtim.tv_sec
#define S_IREAD S_IRUSR
#define S_IWRITE S_IWUSR
#define S_IEXEC S_IXUSR

extern int __fxstat(int __ver, int __fildes, struct stat *
                        *__stat_buf);
extern int __fxstat64(int __ver, int __fildes, struct stat64 *
                          __stat_buf);
extern int __fxstatat(int __ver, int __fildes, const char *
                        *__filename, 
                        struct stat *__stat_buf, int __flag);
extern int __fxstatat64(int __ver, int __fildes, const char *
                         __filename, 
                         struct stat64 *__stat_buf, int __flag);
extern int __lxstat(int __ver, const char * __filename, 
                        struct stat *__stat_buf);
extern int __lxstat64(int __ver, const char * __filename,
struct stat64 *__stat_buf);
extern int __xmknod(int __ver, const char *__path, mode_t __mode,
        dev_t *__dev);
extern int __xmknodat(int __ver, int __fd, const char *__path,
        mode_t __mode, dev_t *__dev);
extern int __xstat(int __ver, const char *__filename,
        struct stat *__stat_buf);
extern int __xstat64(int __ver, const char *__filename,
        struct stat64 *__stat_buf);
extern int chmod(const char *__file, mode_t __mode);
extern int fchmod(int __fd, mode_t __mode);
extern int fchmodat(int __fd, const char *__file, mode_t __mode,
        int __flag);
extern int mkdirat(int __fd, const char *__path, mode_t __mode);
extern int mkfifo(const char *__path, mode_t __mode);
extern int mkfifoat(int __fd, const char *__path, mode_t __mode);
extern mode_t umask(mode_t __mask);

13.4.74 sys/statfs.h

#define NFS_SUPER_MAGIC 0x6969
extern int fstatfs(int __fildes, struct statfs *__buf);
extern int fstatfs64(int __fildes, struct statfs64 *__buf);
extern int statfs(const char *__file, struct statfs *__buf);
extern int statfs64(const char *__file, struct statfs64 *__buf);

13.4.75 sys/statvfs.h

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

13.4.76 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;
}
13.4.77 sys/timeb.h

```c
struct timeb {
    time_t time;                /* Seconds since epoch, as from
        time. */
    unsigned short millitm;     /* Additional milliseconds. */
    short timezone;             /* Minutes west of GMT. */
    short dstflag;              /* Nonzero if Daylight Savings
        Time used. */
};
```

13.4.78 sys/times.h

```c
struct tms {
    clock_t tms_utime;        /* User time used. */
    clock_t tms_stime;        /* System time used. */
    clock_t tms_cutime;       /* CPU time of child. */
    clock_t tms_cstime;       /* CPU time of parent. */
};
```

13.4.79 sys/types.h

```c
#define FALSE   0
#define TRUE    1
#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)   
    ((set)->fds_bits[((d)/(8*sizeof(long)))]=(1<<(d) %8*sizeof(long))))

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 long int blkcnt64_t;
typedef unsigned long long int fsblkcnt64_t;
typedef unsigned long 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;
thedef unsigned long int dev_t;
typedef unsigned int mode_t;
typedef unsigned long int nlink_t;
thedef char *caddr_t;

typedef struct {
    unsigned long int fds_bits[___FDSET_LONGS];
} fd_set;

typedef long int clock_t;
typedef long int time_t;

13.4.80 sys/uio.h

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

13.4.81 sys/un.h

#define UNIX_PATH_MAX   108

struct sockaddr_un {
    sa_family_t sun_family;    /* AF_UNIX */
    char sun_path[UNIX_PATH_MAX];
};

13.4.82 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 *__name);

13.4.83 sys/wait.h

#define WIFSIGNALED(status)      (!WIFSTOPPED(status) && !WEXITED(status))
#define WIFSTOPPED(status)       (((status) & 0xff) == 0x7f)
#define WEXITED(status)          (((status) & 0xff00) >> 8)
#define WTERMSIG(status)         ((status) & 0x7f)
#define WCOREDUMP(status)        ((status) & 0x08)
#define WIFEXITED(status)        (WTERMSIG(status) == 0)
#define WNOHANG 0x00000001
#define WUNTRACED 0x000000002
#define WCOREFLAG 0x80
#define WSTOPSIG(status)         WEXITSTATUS(status)

typedef enum {
    P_ALL,
    P_PID,
    P_PGID
} idtype_t;
extern pid_t wait(int *__stat_loc);
extern pid_t wait4(pid_t __pid, int *__stat_loc, int __options,
                   struct rusage *__usage);
extern int waitid(idtype_t __idtype, id_t __id, siginfo_t *__infop,
                  int __options);
extern pid_t waitpid(pid_t __pid, int *__stat_loc, int __options);

13.4.84 syslog.h

#define LOG_MAKEPRI(fac, pri)  (((fac) << 3) | (pri))
#define LOG_PRI(p)             ((p) & LOG_PRIMASK)     /* extract
    priority */
#define LOG_EMERG 0       /* system is unusable */
#define LOG_PRIMASK 0x07    /* mask to extract priority part */
#define LOG_ALERT 1       /* action must be taken immediately */
#define LOG_CRIT 2       /* critical conditions */
#define LOG_ERR 3        /* error conditions */
#define LOG_WARNING 4     /* warning conditions */
#define LOG_NOTICE 5      /* normal but significant condition */
#define LOG_INFO 6        /* informational */
#define LOG_DEBUG 7       /* debug-level messages */
#define LOG_FAC(p)         (((p) & LOG_FACMASK) >> 3)    /* facility of pri */
#define LOG_KERN (0<<3)    /* kernel messages */
#define LOG_AUTHPRIV (10<<3) /* security/authorization messages (private) */
#define LOG_FTP (11<<3)    /* ftp daemon */
#define LOG_USER (1<<3)    /* random user-level messages */
### ISO/IEC 23360 Part 1:2008(E)

```c
#define LOG_MAIL        (2<<3)  /* mail system */
#define LOG_DAEMON      (3<<3)  /* system daemons */
#define LOG_AUTH        (4<<3)  /* security/authorization messages */
#define LOG_SYSLOG      (5<<3)  /* messages generated internally by syslogd */
#define LOG_LPR (6<<3)          /* line printer subsystem */
#define LOG_NEWS        (7<<3)  /* network news subsystem */
#define LOG_UUCP        (8<<3)  /* UUCP subsystem */
#define LOG_CRON        (9<<3)  /* clock daemon */
#define LOG_FACMASK     0x03f8  /* mask to extract facility part */
#define LOG_LOCAL0      (16<<3) /* reserved for local use */
#define LOG_LOCAL1      (17<<3) /* reserved for local use */
#define LOG_LOCAL2      (18<<3) /* reserved for local use */
#define LOG_LOCAL3      (19<<3) /* reserved for local use */
#define LOG_LOCAL4      (20<<3) /* reserved for local use */
#define LOG_LOCAL5      (21<<3) /* reserved for local use */
#define LOG_LOCAL6      (22<<3) /* reserved for local use */
#define LOG_LOCAL7      (23<<3) /* reserved for local use */
#define LOG_UPTO(pri)   ((1 << ((pri)+1)) - 1)  /* all priorities through pri */
#define LOG_MASK(pri)   (1 << (pri))    /* mask for one priority */
#define LOG_PID 0x01            /* log the pid with each message */
#define LOG_CONS        0x02    /* log on the console if errors in sending */
#define LOG_ODELAY      0x04    /* delay open until first syslog() (default) */
#define LOG_NDELAY      0x08    /* don't delay open */
#define LOG_NOWAIT      0x10    /* don't wait for console forks: DEPRECATED */
#define LOG_PERROR      0x20    /* log to stderr as well */

eextern void __syslog_chk(int, int, const char *, ...);
extern void __vsyslog_chk(int, int, const char *, va_list);
extern void closelog(void);
extern void openlog(const char *__ident, int __option, int __facility);
extern int setlogmask(int __mask);
extern void syslog(int __pri, const char *__fmt, ...);
extern void vsyslog(int __pri, const char *__fmt, va_list __ap);

13.4.85 tar.h

#define REGTYPE '0'
#define LNKTYPE '1'
#define SYMTYPE '2'
#define CHRTYPE '3'
#define BLKTYPE '4'
#define DIRTYPE '5'
#define FIFOTYPE        '6'
#define CONTTYPE        '7'
#define AREGTYPE        '\0'
#define TVERSION        "00"
#define TOEXEC  00001
#define TOWRITE 00002
#define TOREAD  00004
#define TGEXEC  00010
#define TGWRITE 00020
```
13.4.86 termios.h

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

#define NCCS 32

struct termios {
    tcflag_t c_iflag;           /* input mode flags */
    tcflag_t c_oflag;           /* output mode flags */
    tcflag_t c_cflag;           /* control mode flags */
    tcflag_t c_lflag;           /* local mode flags */
    cc_t c_line;                /* line discipline */
    cc_t c_cc[NCCS];            /* control characters */
    speed_t c_ispeed;           /* input speed */
    speed_t c_ospeed;           /* output speed */
};

#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
ISO/IEC 23360 Part 1:2008(E)

#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 B1200   0000010
#define B1800   0000011
#define B4800   0000013
#define B9600   0000015
#define B19200  0000016
#define B38400  0000017

extern speed_t cfgetispeed(const struct termios *__termios_p);
extern speed_t cfgetospeed(const struct termios *__termios_p);
extern void cfmakeraw(struct termios *__termios_p);
extern  int  cfsetispeed(struct  termios  *__termios_p,  speed_t __speed);
extern  int  cfsetospeed(struct  termios  *__termios_p,  speed_t __speed);
extern  int  cfsetspeed(struct  termios  *__termios_p,  speed_t __speed);
extern int tcdrain(int __fd);
extern int tcflow(int __fd, int __action);
extern int tcflush(int __fd, int __queue_selector);
extern int tcgetattr(int __fd, struct termios *__termios_p);
extern pid_t tcgetsid(int __fd);
extern int tcsendbreak(int __fd, int __duration);
extern int tcsetattr(int __fd, int __optional_actions,
    const struct termios *__termios_p);

13.4.87 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;

    © 2008 Linux Foundation
```c
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 *__tp);
extern char *asctime_r(const struct tm *__tp, char *__buf);
extern clock_t clock(void);
extern int clock_getcpuclockid(pid_t __pid, clockid_t *__clock_id);
extern int clock_getres(clockid_t __clock_id, struct timespec *__res);
extern int clock_gettime(clockid_t __clock_id, struct timespec *__tp);
extern int clock_nanosleep(clockid_t __clock_id, int __flags,
    const struct timespec *__req, struct timespec *__rem);
extern int clock_settime(clockid_t __clock_id,
    const struct timespec *__tp);
extern char *ctime(const time_t *__timer);
extern char *ctime_r(const time_t *__timer, char *__buf);
extern int daylight;
extern double difftime(time_t __time1, time_t __time0);
extern struct tm *getdate(const char *__string);
extern int getdate_err;
extern struct tm *gmtime(const time_t *__timer);
extern struct tm *gmtime_r(const time_t *__timer, struct tm *__tp);
extern struct tm *localtime(const time_t *__timer);
extern struct tm *localtime_r(const time_t *__timer, struct tm *__tp);
extern time_t mktime(struct tm *__tp);
extern int nanosleep(const struct timespec *__requested_time,
    struct timespec *__remaining);
extern int stime(const time_t *__when);
extern size_t strftime(char *__s, size_t __maxsize, const char *__format,
    const struct tm *__tp);
extern char *strptime(const char *__s, const char *__fmt, struct tm *__tp);
extern time_t time(time_t *__timer);
extern int timer_create(clockid_t __clock_id, struct sigevent *__evp,
    timer_t *__timerid);
extern int timer_delete(timer_t __timerid);
extern int timer_getoverrun(timer_t __timerid);
extern int timer_gettime(timer_t __timerid, struct itimerspec *__value);
extern int timer_settime(timer_t __timerid, int __flags,
    const struct itimerspec *__value, struct itimerspec *__ovalue);

extern long int timezone;
extern char *tzname[];
extern void tzset(void);

13.4.88 ucontext.h
```
extern int getcontext(ucontext_t *__ucp);
extern void makecontext(ucontext_t *__ucp, void (*__func)(void),
                        int __argc, ...);
extern int setcontext(const struct ucontext *__ucp);
extern int swapcontext(ucontext_t *__oucp, const struct ucontext *__ucp);

13.4.89 ulimit.h

#define UL_GETFSIZE     1
#define UL_SETFSIZE     2

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

13.4.90 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 _POSIX2_C_BIND  200112L
#define _POSIX2_VERSION 200112L
#define _POSIX_THREADS  200112L
#define _POSIX_VERSION  200112L

#define _PC_LINK_MAX    0
#define _PC_MAX_CANON   1
#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
#define _SC_CHAR_BIT 101
#define _SC_CHAR_MAX 102
#define _SC_CHAR_MIN 103
#define _SC_SHRT_MAX 104
#define _SC_INT_MAX 105
#define _SC_LONG_BIT 106
#define _SC_SHRT_MIN 107
#define _SC_NZERO 108
#define _SC_TIMERS 11
#define _SC_SSIZE_MAX 110
#define _SC_SCHAR_MAX 111
#define _SC_SCHAR_MIN 112
#define _SC_SCHAR_MAX 113
#define _SC_SCHAR_MIN 114
#define _SC_UCHAR_MAX 115
#define _SC_SINT_MAX 116
#define _SC_SLONG_MAX 117
#define _SC_USHRT_MAX 118
#define _SC_NL_ARGMAX 119
#define _SC_ASYNC_IO 120
#define _SC_NL_LANGMAX 121
#define _SC_NL_MSGMAX 122
#define _SC_NL_NMAX 123
#define _SC_NL_TEXTMAX 124
#define _SC_XBS5_ILP32_OFF32 125
#define _SC_XBS5_ILP32_OFFBIG 126
#define _SC_XBS5_LP64_OFF64 127
#define _SC_XBS5_LPBIG_OFFBIG 128
#define _SC_XOPEN_LEGACY 129
#define _SC_XOPEN_REALTIME 130
#define _SC_XOPEN_REALTIME_THREADS 131
#define _SC_ADVISORY_INFO 132
#define _SC_BARRIERS 133
#define _SC_BASE 134
#define _SC_CLOCK_SELECTION 135
#define _SC_CLOCK_SELECTION 136
#define _SC_CPUTIME 137
#define _SC_THREAD_CPUTIME 138
#define _SC_SYNCHRONIZED_IO 139
#define _SC_DEVICE_I0 140
#define _SC_DEVICE_SPECIFIC 141
#define _SC_DEVICE_SPECIFIC_R 142
#define _SC_FD_MGMT 143
#define _SC_FIFO 144
#define _SC_PIPE 145
#define _SC_FILE_ATTRIBUTES 146
#define _SC_FILE_LOCKING 147
#define _SC_FILE_SYSTEM 148
#define _SC_MONOTONIC_CLOCK 149
#define _SC_FSYNC 150
#define _SC_MULTI_PROCESS 151
#define _SC_NETWORKING 152
#define _SC_READER_WRITER_LOCKS 153
#define _SC_SPIN_LOCKS  154
#define _SC_REGEX_LOCKS  155
#define _SC_REGEX_VERSION       156
#define _SC_SHELL       157
#define _SC_SIGNALS     158
#define _SC_SPAWN       159
#define _SC_MAPPED_FILES        16
#define _SC_SPORADIC_SERVER   160
#define _SC_THREAD_SPORADIC_SERVER  161
#define _SC_SYSTEM_DATABASE 162
#define _SC_SYSTEM_DATABASE_R  163
#define _SC_TIMEOUTS    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_PBSACCOUNTING 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_SYMLOOP_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_LPBIG_OFFBIG 179
#define _SC_MEMLOCK_RANGE  18
#define _SC_HOST_NAME_MAX 180
#define _SC_TRACE       181
#define _SC_TRACEEVENT_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_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_NGROUPS_MAX 3
#define _SC_PAGESIZE 30

© 2008 Linux Foundation
#define _SC_PAGE_SIZE   30
#define _SC_RTSIG_MAX   31
#define _SC_SEM_NSEMS_MAX       32
#define _SC_SEM_VALUE_MAX       33
#define _SC_TIMER_MAX   35
#define _SC_BC_BASE_MAX 36
#define _SC_BC_DIM_MAX  37
#define _SC_BC_SCALE_MAX        38
#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_POLL 58
#define _SC_SELECT 59
#define _SC_TZNAME_MAX  6
#define _SC_IOV_MAX     60
#define _SC_UIO_MAXIOV  60
#define _SC_PII_INTERNET_STREAM 61
#define _SC_PII_INTERNET_DGRAM 62
#define _SC_PII_OSI_COTS 63
#define _SC_PII_OSI_CLTS 64
#define _SC_PII_OSI_M  65
#define _SC_T_IOV_MAX   66
#define _SC_THREADS    67
#define _SC_THREAD_SAFE_FUNCTIONS 68
#define _SC_GETGR_R_SIZE_MAX    69
#define _SC_JOB_CONTROL 7
#define _SC_GETPW_R_SIZE_MAX    70
#define _SC_LOGIN_NAME_MAX      71
#define _SC_TTY_NAME_MAX        72
#define _SC_THREAD_DESTRUCTOR_ITERATIONS 73
#define _SC_THREAD_KEYS_MAX     74
#define _SC_THREAD_STACK_MIN    75
#define _SC_THREAD_THREADS_MAX  76
#define _SC_THREAD_ATTR_STACKADDR       77
#define _SC_THREAD_ATTR_STACKSIZE       78
#define _SC_THREAD_PRIORITY_SCHEDULING 79
#define _SC_SAVED_IDS   8
#define _SC_THREAD_PRIO_INHERIT 80
#define _SC_THREAD_PRIO_PROTECT 81
#define _SC_THREAD_PROCESS_SHARED       82
#define _SC_NPROCESSORS_CONF    83
#define _SC_NPROCESSORS_ONLN    84
#define _SC_PHYS_PAGES  85
#define _SC_AVPHYS_PAGES        86
#define _SC_ATEXIT_MAX  87
#define _SC_PASS_MAX    88
#define _SC_OPEN_VERSION 89
#define _SC_REALTIME_SIGNALS 9
#define _SC_XOPEN_XCU_VERSION 90
#define _SC_XOPEN_UNIX 91
#define _SC_XOPEN_CRYPT 92
#define _SC_XOPEN_ENH_I18N 93
#define _SC_XOPEN_SHM 94
#define _SC_2_CHAR_TERM 95
#define _SC_2_C_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 XOPEN_VERSION 500

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

extern size_t __confstr_chk(int, char *, size_t, size_t);
extern char **__environ;
extern char *__environ;
extern char *__getcwd_chk(char *, size_t, size_t);
extern int __getgroups_chk(int, gid_t *, size_t);
extern int __getgroups_chk(int, const gid_t *, size_t);
extern int __gethostname_chk(char *, size_t, size_t);
extern int __getlogin_r_chk(char *, size_t, size_t);
extern pid_t __getpgid(pid_t __pid);
extern ssize_t __pread64_chk(int, void *, size_t, off64_t, size_t);
extern ssize_t __pread_chk(int, void *, size_t, off_t, size_t);
extern ssize_t __read_chk(int, void *, size_t, size_t);
extern ssize_t __readlink_chk(const char *, char *, size_t, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_chk(const char *, const char *, size_t);
extern ssize_t __readlink_ch...
extern char *ctermid(char *__s);
extern char *cuserid(char *__s);
extern int daemon(int __nochdir, int __noclose);
extern int dup(int __fd);
extern int dup2(int __fd, int __fd2);
extern void encrypt(char *__block, int __edflag);
extern int exec1(const char *__path, const char *__arg, ...);
extern int execle(const char *__path, const char *__arg, ...);
extern int exec1p(const char *__file, const char *__arg, ...);
extern int execve(const char *__path, const char *__argv[],
                    char *__envp[]);
extern int execvp(const char *__file, char *__argv[]);
extern int faccessat(int __fd, const char *__file, int __type,
                     int __flag);
extern int fchdir(int __fd);
extern int fchown(int __fd, uid_t __owner, gid_t __group);
extern int fchownat(int __fd, const char *__file, uid_t __owner,
                    gid_t __group, int __flag);
extern int fdatasync(int __fildes);
extern int fexecve(int __fd, char *const __argv[], char *const __envp[]);
extern pid_t fork(void);
extern long int fpathconf(int __fd, int __name);
extern int fsync(int __fd);
extern int ftruncate(int __fd, off_t __length);
extern int ftruncate64(int __fd, off64_t __length);
extern int getcwd(char *__buf, size_t __size);
extern int getdomainname(char *__name, size_t __len);
extern int getdtablesize(void);
extern gid_t getegid(void);
extern uid_t geteuid(void);
extern gid_t getgid(void);
extern int getgroups(int __size, gid_t __list[]);
extern long int gethostname(char *__name, size_t __len);
extern int getlogin(void);
extern int getlogin_r(char *__name, size_t __name_len);
extern int getopt(int ___argc, char *const ___argv[],
                   const char *__shortopts);
extern int getpagesize(void);
extern pid_t getpgid(pid_t __pid);
extern pid_t getpgrp(void);
extern pid_t getpid(void);
extern pid_t getppid(void);
extern pid_t getsid(pid_t __pid);
extern uid_t getuid(void);
extern int isatty(int __fd);
extern int link(const char *__from, const char *__to);
extern int linkat(int __fromfd, const char *__from, int __tofd,
                  const char *__to, int __flags);
extern int lockf(int __fd, int __cmd, off_t __len);
extern int lockf64(int __fd, int __cmd, off64_t __len);
extern int lseek(int __fd, off_t __offset, int __whence);
extern int lseek64(int __fd, loff_t __offset, int __whence);
extern int mkdir(const char *__path, mode_t __mode);
extern int mkstemp(char *__template);
extern int nice(int __inc);
extern int optarg;
extern int opterr;
extern int optind;
extern int optopt;
extern long int pathconf(const char *__path, int __name);
extern int pause(void);
extern int pipe(int __pipedes[2]);
extern ssize_t pread(int __fd, void *__buf, size_t __nbytes,
  off_t __offset);
extern ssize_t pread64(int __fd, void *__buf, size_t __nbytes,
  off64_t __offset);
extern ssize_t pwrite(int __fd, const void *__buf, size_t __n,
  off_t __offset);
extern ssize_t pwrite64(int __fd, const void *__buf, size_t __n,
  off64_t __offset);
extern ssize_t read(int __fd, void *__buf, size_t __nbytes);
extern ssize_t readlink(const char *__path, char *__buf, size_t __len);
extern ssize_t readlinkat(int __fd, const char *__path, char *__buf,
  size_t __len);
extern int rename(const char *__old, const char *__new);
extern int rmdir(const char *__path);
extern int setegid(gid_t __gid);
extern int seteuid(uid_t __uid);
extern int setgid(gid_t __gid);
extern int sethostname(const char *__name, size_t __len);
extern int setkey(const char *__key);
extern int setpgid(pid_t __pid, pid_t __pgid);
extern int setpgrp(void);
extern int setregid(gid_t __rgid, gid_t __egid);
extern int setreuid(uid_t __ruid, uid_t __euid);
extern int setuid(uid_t __uid);
extern int setuid_r(int __fd, char *__buf, size_t __buflen);
extern unsiged int sleep(unsigned int __seconds);
extern int symlink(const char *__from, const char *__to);
extern int symlinkat(const char *__from, int __tofd, const char *__to);
extern void sync(void);
extern unsigned int ualarm(useconds_t __value, useconds_t __interval);
extern int unlink(const char *__name);
extern int unlinkat(int __fd, const char *__name, int __flag);
extern int usleep(useconds_t __useconds);
extern pid_t vfork(void);
extern ssize_t write(int __fd, const void *__buf, size_t __n);

13.4.91 utime.h

struct utimbuf {
    time_t actime;
    time_t modtime;
};
extern int utime(const char *__file, const struct utimbuf *__file_times);
13.4.92 utmp.h

#define UT_HOSTSIZE  256
#define UT_LINESIZE  32
#define UT_NAMESIZE  32
#define ut_addr ut_addr_v6[0]
#define ut_time ut_tv.tv_sec
#define ut_name ut_user /* Backwards compatability */

struct exit_status {
  short e_termination; /* Process termination status. */
  short e_exit; /* Process exit status. */
};

#define EMPTY   0 /* No valid user accounting information. */
#define RUN_LVL  1 /* The system's runlevel. */
#define BOOT_TIME  2 /* Time of system boot. */
#define NEW_TIME  3 /* Time after system clock changed. */
#define OLD_TIME  4 /* Time when system clock changed. */
#define INIT_PROCESS  5 /* Process spawned by the init process. */
#define LOGIN_PROCESS  6 /* Session leader of a logged in user. */
#define USER_PROCESS  7 /* Normal process. */
#define DEAD_PROCESS  8 /* Terminated process. */
#define ACCOUNTING  9

extern void endutent(void);
extern struct utmp *getutent(void);
extern int getutent_r(struct utmp *__buffer, struct utmp *__result);
extern void login(const struct utmp *__entry);
extern int login_tty(int __fd);
extern int logout(const char *__ut_line);
extern void logwtmp(const char *__ut_line, const char *__ut_name, const char *__ut_host);
extern void setutent(void);
extern int utmpname(const char *__file);

13.4.93 utmpx.h

extern void endutxent(void);
extern struct utmpx *getutxent(void);
extern struct utmpx *getutxid(const struct utmpx *__id);
extern struct utmpx *getutxline(const struct utmpx *__line);
extern struct utmpx *pututxline(const struct utmpx *__utmpx);
extern void setutxent(void);
extern int utmpxname(const char *__file);

13.4.94 wchar.h

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

extern wchar_t *__fgetws_chk(wchar_t *, size_t, int, FILE *);
extern wchar_t  *__fgetws_unlocked_chk(wchar_t *, size_t, int, FILE *);
extern int __fwprintf_chk(FILE *, int, const wchar_t *, ...);
extern size_t __mbsntowcs_chk(wchar_t *, const char **, size_t, mbstate_t *, size_t);
extern size_t __mbsrtowcs_chk(wchar_t *, const char **, size_t, mbstate_t *, size_t);
extern int __swprintf_chk(wchar_t *, size_t, int, size_t, const wchar_t *, ...
);extern int __vfwprintf_chk(FILE *, int, const wchar_t *, va_list);
extern int __vswprintf_chk(wchar_t *, size_t, int, size_t, const wchar_t *, va_list);
extern int __vwprintf_chk(int, const wchar_t *, va_list);
extern wchar_t *__wcpcpy_chk(wchar_t *, const wchar_t *, size_t);
extern wchar_t *__wcpncpy_chk(wchar_t *, const wchar_t *, size_t, size_t);
extern size_t __wcrtomb_chk(char *, wchar_t, mbstate_t *, size_t);
extern wchar_t *__wcscat_chk(wchar_t *, const wchar_t *, size_t);
extern wchar_t *__wcsncpy_chk(wchar_t *, const wchar_t *, size_t, size_t);
extern wchar_t *__wcsncat_chk(wchar_t *, const wchar_t *, size_t, size_t);
extern size_t __wcsnrtombs_chk(char *, const wchar_t * *, size_t, mbstate_t *, size_t);
extern size_t __wcsrtombs_chk(char *, const wchar_t * *, size_t, mbstate_t *, size_t);
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 *__wmemcpy_chk(wchar_t *, const wchar_t *, size_t, size_t);
extern wchar_t *__wmemmove_chk(wchar_t *, const wchar_t *, size_t, size_t);
extern wchar_t *__wmempcpy_chk(wchar_t *, const wchar_t *, size_t, size_t);
extern wchar_t *__wmemset_chk(wchar_t *, wchar_t, size_t, size_t);
extern int __wprintf_chk(int, const wchar_t *, ...);
extern wint_t btowc(int __c);
extern wint_t fgetwc(FILE * __stream);
extern wint_t fgetws(FILE * __stream, int __n);
extern wint_t fgetwc_unlocked(FILE * __stream);
extern wchar_t *fgetws_unlocked(FILE * __stream, int __n);
extern wint_t fputwc(wchar_t __wc, FILE * __stream);
extern wint_t fputwc_unlocked(wchar_t __wc, FILE * __stream);
extern int fputws(const wchar_t *__ws, FILE * __stream);
extern int fputws_unlocked(const wchar_t *__ws, FILE * __stream);
extern void fwide(FILE * __fp, int __mode);
extern int fwprintf(FILE * __stream, const wchar_t * __format, ...);extern int fwscanf(FILE * __stream, const wchar_t * __format, ...);
extern wint_t getwc(FILE *__stream);
extern wint_t getwc_unlocked(FILE *__stream);
extern wint_t getwchar(void);
extern wint_t getwchar_unlocked(void);
extern size_t mbrlen(const char *__s, size_t __n, mbstate_t *__ps);
extern size_t mbtowc(wchar_t *__pwc, const char *__s, size_t __n, mbstate_t *__p);
extern int mbsinit(const mbstate_t *__ps);
extern size_t mbsnrtowcs(wchar_t *__dst, const char *__s, size_t __nmc, size_t __len, mbstate_t *__ps);
extern size_t mbsrtowcs(wchar_t *__dst, const char *__s, size_t __len, mbstate_t *__ps);
extern FILE *open_wmemstream(wchar_t * *__bufloc, size_t *__sizeloc);
extern wint_t putwc(wchar_t __wc, FILE *__stream);
extern wint_t putwc_unlocked(wchar_t __wc, FILE *__stream);
extern wint_t putwchar(wchar_t __wc);
extern wint_t putwchar_unlocked(wchar_t __wc);
extern int swprintf(wchar_t * __s, size_t __n, const wchar_t * __format, ...);
extern int swscanf(const wchar_t * __s, const wchar_t * __format, va_list __arg);
extern int vfwprintf(FILE * __s, const wchar_t * __format, va_list __arg);
extern int vfwscanf(FILE * __s, const wchar_t * __format, va_list __arg);
extern int vswprintf(wchar_t * __s, size_t __n, const wchar_t * __format, va_list __arg);
extern int vswscanf(const wchar_t * __s, const wchar_t * __format, va_list __arg);
extern wchar_t *wcpcpy(wchar_t * __dest, const wchar_t * __src);
extern wchar_t *wcpncpy(wchar_t * __dest, const wchar_t * __src, size_t __n);
extern size_t wcrtomb(char *__s, wchar_t __wc, mbstate_t *__ps);
extern int wcscasecmp(const wchar_t * __s1, const wchar_t * __s2);
extern wchar_t *wcsncpy(wchar_t * __dest, const wchar_t * __src, size_t __n);
extern int wcscmp(const wchar_t * __s1, const wchar_t * __s2);
extern wchar_t *wcsncat(wchar_t * __dest, const wchar_t * __src, size_t __n);
extern int wcscoll(const wchar_t * __s1, const wchar_t * __s2);
extern size_t wcsmsprint(const wchar_t * __wcsp, const wchar_t * __wc, const wchar_t * __reject);
extern wchar_t *wcsdup(const wchar_t * __s);
extern size_t wcsftime(wchar_t * __s, size_t __maxsize, const wchar_t * __format, const struct tm *__tp);
extern size_t wcslen(const wchar_t * __s);
extern int wcscascmpeq(const wchar_t * __s1, const wchar_t * __s2, size_t __n);
extern wchar_t *wcsncat(wchar_t * __dest, const wchar_t * __src, size_t __n);
extern int wcscncmp(const wchar_t * __s1, const wchar_t * __s2, size_t __n);
extern wchar_t *wcsncpy(wchar_t * __dest, const wchar_t * __src, size_t __n);
include \nstruct

size_t __n);
extern size_t wcsnlen(const wchar_t *__s, size_t __maxlen);

extern size_t wcnrtombs(char *__dst, const wchar_t * *__src,
size_t __nwc,
size_t __len, mbstate_t *__ps);

extern wchar_t *wcschr(const wchar_t *__wcs, wchar_t __wc);
extern wchar_t *wcspbrk(const wchar_t *__wcs, const wchar_t *__accept);

extern size_t wcsrtombs(char *__dst, const wchar_t * *__src,
size_t __len,
mbstate_t *__ps);

extern wctype_t *wctype(const wchar_t *__c);

extern wchar_t *wcscnlen(const wchar_t *__s, size_t __maxlen);

extern wchar_t *wcscsntomb(char *__dst, const wchar_t * *__src,
size_t __nwc,
size_t __len, mbstate_t *__ps);

extern wchar_t *wcscstr(const wchar_t *__haystack,
const wchar_t *__needle);

extern size_t wcsxfrm(wchar_t * __s1, const wchar_t * __s2,
size_t __n);

extern wchar_t *wmemchr(const wchar_t * __s, wchar_t __c, size_t __n);
extern int wmemcmp(const wchar_t * __s1, const wchar_t * __s2,
size_t __n);

extern wchar_t *wmemcpy(wchar_t * __s1, const wchar_t * __s2,
size_t __n);
extern wchar_t *wmemset(wchar_t * __s, wchar_t __c, size_t __n);

extern int wprintf(const wchar_t * __format, ...);
extern int wscanf(const wchar_t * __format, ...);

13.4.95 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 iswalnum(wint_t __wc);
extern int iswalpha(wint_t __wc);
extern int iswblank(wint_t __wc);
extern int iswcntrl(wint_t __wc);
extern int iswctype(wint_t __wc, wctype_t __desc);
extern int iswdigit(wint_t __wc);
extern int iswgraph(wint_t __wc);
extern int iswlower(wint_t __wc);
extern int iswprint(wint_t __wc);
extern int iswpunct(wint_t __wc);
extern int iswspace(wint_t __wc);
extern int iswupper(wint_t __wc);
extern int iswxdigit(wint_t __wc);
extern wint_t towctrans(wint_t __wc, wctrans_t __desc);
extern wint_t towlower(wint_t __wc);
extern wint_t towupper(wint_t __wc);
extern wctrans_t wctrans(const char *__property);
extern wctype_t wctype(const char *__property);

13.4.96 wordexp.h

enum {
    WRDE_DOFFS = 1,
    WRDE_APPEND = 2,
    WRDE_NOCMD = 4,
    WRDE_REUSE = 8,
    WRDE_SHOWERR = 16,
    WRDE_UNDEF = 32
};

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

enum {
    WRDE_NOSYS = -1,
    WRDE_NOSPACE = 1,
    WRDE_BADCHAR = 2,
    WRDE_BADVAL = 3,
    WRDE_CMDSUB = 4,
    WRDE_SYNTAX = 5
};

extern int wordexp(const char *__words, wordexp_t *__pwordexp,
                    int __flags);
extern void wordfree(wordexp_t * __wordexp);

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
_IOC_feof — alias for feof

Synopsis
int _IO_feof(_IO_FILE * __fp);

Description
_IOC_feof() tests the end-of-file indicator for the stream pointed to by __fp, returning a non-zero value if it is set. _IOC_feof() is not in the source standard; it is only in the binary standard.

_IOC_getc

Name
_IOC_getc — alias for getc

Synopsis
int _IO_getc(_IO_FILE * __fp);

Description
_IOC_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. _IOC_getc() is not in the source standard; it is only in the binary standard.

_IOC_putchar

Name
_IOC_putchar — alias for putc

Synopsis
int _IO_putchar(int __c, _IO_FILE * __fp);

Description
_IOC_putchar() writes the character __c, cast to an unsigned char, to __fp. _IOC_putchar() is not in the source standard; it is only in the binary standard.
_IO_puts

Name

_IO_puts — alias for puts

Synopsis

int _IO_puts(const char * __c);

Description

_IO_puts() writes the string __s and a trailing newline to stdout.
_IO_puts() is not in the source standard; it is only in the binary standard.

__assert_fail

Name

__assert_fail — abort the program after false assertion

Synopsis

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

Description

The __assert_fail() function is used to implement the assert() interface of ISO POSIX (2003). The __assert_fail() function shall print the given file filename, line line number, function function name and a message on the standard error stream in an unspecified format, and abort program execution via the abort() function. For example:
   a.c:10: foobar: Assertion a == b failed.
If function is NULL, __assert_fail() shall omit information about the function.
assertion, file, and line shall be non-NULL.
The __assert_fail() function is not in the source standard; it is only in the binary standard. The assert() interface is not in the binary standard; it is only in the source standard. The assert() may be implemented as a macro.
__chk_fail

Name
__chk_fail — terminate a function in case of buffer overflow

Synopsis
#include <libc.h>
void __chk_fail(void);

Description
The interface __chk_fail() shall abort the function that called it with a message that a buffer overflow has been detected. The program that called the function shall then exit.

Application Usage (informative)
The interface __chk_fail() does not check for a buffer overflow itself. It merely reports one when invoked.

__confstr_chk

Name
__confstr_chk — get configuration dependent string variables, with buffer overflow checking

Synopsis
#include <unistd.h>
size_t __confstr_chk(int name, char * buf, size_t len, size_t buflen);

Description
The interface __confstr_chk() shall function in the same way as the interface confstr(), except that __confstr_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the buffer buf. If len exceeds buflen, the function shall abort, and the program calling it shall exit.

The __confstr_chk() function is not in the source standard; it is only in the binary standard.
__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 cur-
rent 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 char-
acter in the current locale.

__ctype_get_mb_cur_max() is not in the source standard; it is only in the bi-
nary standard.
__ctype_tolower_loc

Name
__ctype_tolower_loc — accessor function for __ctype_b_tolower array for
cctype 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
caracters in the current locale that contains lower case equivalents for each
caracter in the current character set. The array shall contain a total of 384 char-
acters, 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
caracters to be used for the cctype() family of functions (see <ctype.h>.

__ctype_toupper_loc

Name
__ctype_toupper_loc — accessor function for __ctype_b_toupper() array
for cctype 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
caracters in the current locale that contains upper case equivalents for each
caracter in the current character set. The array shall contain a total of 384 char-
acters, 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
caracters to be used for the cctype() family of functions (see <ctype.h>.

© 2008 Linux Foundation
__cxa_atexit

Name

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

Synopsis

```c
int __cxa_atexit(void (*) (void *), void *, void *, 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**

```c
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 — external daylight savings time flag

Synopsis
int __daylight;

Description
The external variable __daylight shall implement the daylight savings time flag daylight as specified in ISO POSIX (2003). __daylight has the same specification as daylight.

__environ

Name
__environ — alias for environ - user environment

Synopsis
extern char **__environ;

Description
The external variable __environ shall implement the environment variable environ as specified in ISO POSIX (2003). __environ has the same specification as environ.

__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.
__fgets_chk

Name
__fgets_chk — string input, with buffer overflow checking

Synopsis
#include <stdio.h>
char * __fgets_chk(char * s, size_t size, int strsize, FILE * stream);

Description
The interface __fgets_chk() shall function in the same way as the interface fgets(), except that __fgets_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter strsize specifies the size of the object pointed to by stream.

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

__fgets_unlocked_chk

Name
__fgets_unlocked_chk — non-locking string input, with buffer overflow checking

Synopsis
#include <stdio.h>
char * __fgets_unlocked_chk(char * s, size_t size, int strsize, FILE * stream);

Description
The interface __fgets_unlocked_chk() shall function in the same way as the interface fgets_unlocked(), except that __fgets_unlocked_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter strsize specifies the size of the object pointed to by stream.

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

Name

__fgetws_chk — read a wide-character string from a FILE stream, with buffer overflow checking

Synopsis

#include <wchar.h>
wchar_t * __fgetws_chk(wchar_t * ws, size_t size, int strsize, FILE * stream);

Description

The interface __fgetws_chk() shall function in the same way as the interface fgetws(), except that __fgetws_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter strsize specifies the size of the object pointed to by stream.

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

__fgetws_unlocked_chk

Name

__fgetws_unlocked_chk — read a wide-character string from a FILE stream in a non-locking manner, with stack checking

Synopsis

#include <wchar.h>
wchar_t * __fgetws_unlocked.chk(wchar_t * ws, size_t strsize, int n, FILE * stream);

Description

The interface __fgetws_unlocked_chk() shall function in the same way as the interface fgetws_unlocked(), except that __fgetws_unlocked_chk() shall check for stack overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter strsize specifies the size of the object pointed to by stream.

The __fgetws_unlocked_chk() function 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
def __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.

__fprintf_chk

Name
__fprintf_chk — convert formatted output, with stack checking

Synopsis
#include <libc.h>
def __fprintf_chk(FILE * stream, int flag, const char * format);

Description
The interface __fprintf_chk() shall function in the same way as the interface
fprintf(), except that __fprintf_chk() shall check for stack overflow before
computing a result, depending on the value of the flag parameter. If an over-
flow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this inter-
face shall take in the form of checking the stack, parameter values, and so on.

The __fprintf_chk() function is not in the source standard; it is only in the bi-
nary standard.
__fwprintf_chk

Name
__fwprintf_chk — convert formatted wide-character output, with stack checking

Synopsis
#include <wchar.h>
int __fwprintf_chk(FILE * stream, int flag, const wchar_t * format);

Description
The interface __fwprintf_chk() shall function in the same way as the interface fwprintf(), except that __fwprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

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

__fxstatat

Name
__fxstatat — get file status relative to directory file descriptor

Synopsis
#include <fcntl.h>
#include <sys/stat.h>
int __fxstatat(int ver, int dirfd, const char * path, struct stat * stat_buf, int flags);

Description
The __fxstatat() function shall implement the fstatat() function. The behavior of __fxstatat() for values of ver other than _STAT_VER is undefined. See Data Definitions in the architecture specific part of this specification for the correct value of _STAT_VER.

__fxstatat(_STAT_VER, dirfd, stat_buf, flags) shall behave as fstatat(dirfd, stat_buf, flags) as specified by POSIX 1003.1 2008.

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

Note: The fstatat() function is not in the binary standard; it is only in the source standard.
__fxstatat64

Name

__fxstatat64 — get file status relative to directory file descriptor

Synopsis

#define __LARGEFILE_SOURCE 1
#include <fcntl.h>
#include <sys/stat.h>
int __fxstatat64(int ver, int dirfd, const char * path, struct stat64 * stat_buf, int flags);

Description

The __fxstatat64() function shall implement the fstatat64() function. The behavior of __fxstatat64() for values of ver other than _STAT_VER is undefined. See Data Definitions in the architecture specific part of this specification for the correct value of _STAT_VER.

__fxstatat64(_STAT_VER, dirfd, stat_buf, flags) shall behave as fstatat64(dirfd, stat_buf, flags) as specified by this specification.

__fxstatat64() is not in the source standard; it is only in the binary standard.

Note: The fstatat64() function is not in the binary standard; it is only in the source standard.

__getcwd_chk

Name

__getcwd_chk — get current working directory, with buffer overflow checking

Synopsis

#include <unistd.h>
char * __getcwd_chk(char * buf, size_t len, size_t buflen);

Description

The interface __getcwd_chk() shall function in the same way as the interface getcwd(), except that __getcwd_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the buffer buf. If len exceeds buflen, the function shall abort, and the program calling it shall exit.

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

Name

__getgroups_chk — get list of supplementary group IDs, with buffer
overflow checking

Synopsis

#include <unistd.h>
int __getgroups_chk(int size, gid_t * list, size_t listlen);

Description

The interface __getgroups_chk() shall function in the same way as the inter-
face getgroups(), except that __getgroups_chk() shall check for buffer over-
flow before computing a result. If an overflow is anticipated, the function shall
abort and the program calling it shall exit.

The parameter listlen specifies the size in bytes of the object list.

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

__gethostname_chk

Name

__gethostname_chk — get host name, with buffer overflow checking

Synopsis

#include <unistd.h>
int __gethostname_chk(char * buf, size_t buflen, size_t maxlen);

Description

The interface __gethostname_chk() shall function in the same way as the inter-
face gethostname(), except that __gethostname_chk() shall check for buffer over-
flow before computing a result. If an overflow is anticipated, the function shall
abort and the program calling it shall exit.

The parameter buflen specifies the size of the buffer buf. If buflen exceeds
maxlen, the function shall abort, and the program calling it shall exit.

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

Name
__getlogin_r_chk — get user name, with buffer overflow checking (reentrant)

Synopsis
#include <unistd.h>
int __getlogin_r_chk(char *buf, size_t buflen, size_t maxlen);

Description
The interface __getlogin_r_chk() shall function in the same way as the interface getlogin_r(), except that __getlogin_r_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the buffer buf. If buflen exceeds maxlen, the function shall abort, and the program calling it shall exit.

The __getlogin_r_chk() function 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**

```c
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**

```c
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**

```c
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 ** ubp_av, 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.
__mbsnrtowcs_chk

Name

__mbsnrtowcs_chk — convert a multibyte string to a wide-character string, with buffer overflow checking

Synopsis

#include <wchar.h>
size_t __mbsnrtowcs_chk(wchar_t * dest, const char * * src, size_t nmc, size_t len, mbstate_t * ps, size_t destlen);

Description

The interface __mbsnrtowcs_chk() shall function in the same way as the interface mbsnrtowcs(), except that __mbsnrtowcs_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds destlen, the function shall abort, and the program calling it shall exit.

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

__mbsrtowcs_chk

Name

__mbsrtowcs_chk — convert a multibyte string to a wide-character string, with buffer overflow checking

Synopsis

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

Description

The interface __mbsrtowcs_chk() shall function in the same way as the interface mbsrtowcs(), except that __mbsrtowcs_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds destlen, the function shall abort, and the program calling it shall exit.

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

Name

__mbstowcs_chk — convert a multibyte string to a wide-character string, with buffer overflow checking

Synopsis

#include <stdlib.h>
size_t __mbstowcs_chk(wchar_t * dest, const char * src, size_t len, size_t destlen);

Description

The interface __mbstowcs_chk() shall function in the same way as the interface mbstowcs(), except that __mbstowcs_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds destlen, the function shall abort, and the program calling it shall exit.

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

__memcpy_chk

Name

__memcpy_chk — copy memory area, with buffer overflow checking

Synopsis

#include <string.h>
void * __memcpy_chk(void * dest, const void * src, size_t len, size_t destlen);

Description

The interface __memcpy_chk() shall function in the same way as the interface memcpy(), except that __memcpy_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds destlen, the function shall abort, and the program calling it shall exit.

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

Name

__memmove_chk — copy memory area, with buffer overflow checking

Synopsis

#include <string.h>
void * __memmove_chk(void * dest, const void * src, size_t len, size_t destlen);

Description

The interface __memmove_chk() shall function in the same way as the interface memmove(), except that __memmove chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds destlen, the function shall abort, and the program calling it shall exit.

The __memmove_chk() function 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.

If copying takes place between objects that overlap, the behavior is undefined.

If either dest or src is a null pointer, the behavior is undefined.

If n is 0 and the other parameters are valid, the return value is dest.

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

Name

__mempcpy_chk — copy memory area, with buffer overflow checking

Synopsis

#include <string.h>

void *
__mempcpy_chk(void * dest, const void * src, size_t len, size_t destlen);

Description

The interface __mempcpy_chk() shall function in the same way as the interface
mempcpy(), except that __mempcpy_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds
destlen, the function shall abort, and the program calling it shall exit.

The __mempcpy_chk() function is not in the source standard; it is only in the bi-
nary standard.

__memset_chk

Name

__memset_chk — fill memory with a constant byte, using buffer overflow
checking

Synopsis

#include <string.h>

void *
__memset_chk(void * dest, int c, size_t len, size_t destlen);

Description

The interface __memset_chk() shall function in the same way as the interface
memset(), except that __memset_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object dest. If len exceeds
destlen, the function shall abort, and the program calling it shall exit.

The __memset_chk() function is not in the source standard; it is only in the bi-
nary standard.
__pread64_chk

Name

__pread64_chk — read from a file descriptor at a given offset, with buffer overflow checking

Synopsis

#include <unistd.h>
ssize_t __pread64_chk(int fd, void * buf, size_t nbytes, off64_t offset, size_t buflen);

Description

The interface __pread64_chk() shall function in the same way as the interface pread64(), except that __pread64_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the buffer buf. If nbytes exceeds buflen, the function shall abort, and the program calling it shall exit.

The __pread64_chk() function is not in the source standard; it is only in the binary standard.

__pread_chk

Name

__pread_chk — read from a file descriptor at a given offset, with buffer overflow checking

Synopsis

#include <unistd.h>
ssize_t __pread_chk(int fd, void * buf, size_t nbytes, off_t offset, size_t buflen);

Description

The interface __pread_chk() shall function in the same way as the interface pread(), except that __pread_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the buffer buf. If nbytes exceeds buflen, the function shall abort, and the program calling it shall exit.

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

Name

__printf_chk — format and print data, with stack checking

Synopsis

#include <stdio.h>
int __printf_chk(int flag, const char * format);

Description

The interface __printf_chk() shall function in the same way as the interface printf(), except that __printf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

The __printf_chk() function 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

The __rawmemchr() function shall locate the first occurrence of c (converted to an unsigned char) in the object pointed to by s. If the byte does not occur in the object, then the behavior is undefined.

__rawmemchr() is a weak alias for 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.

Return Value

The __rawmemchr() function shall return a pointer to the located byte.
__read_chk

Name
__read_chk — read from a file descriptor, with buffer overflow checking

Synopsis
#include <unistd.h>
ssize_t __read_chk(int fd, void * buf, size_t nbytes, size_t buflen);

Description
The interface __read_chk() shall function in the same way as the interface
read(), except that __read_chk() shall check for buffer overflow before com-
puting a result. If an overflow is anticipated, the function shall abort and the
program calling it shall exit.
The parameter buflen specifies the size of the buffer buf. If nbytes exceeds bu-
flen, the function shall abort, and the program calling it shall exit.
The __read_chk() function is not in the source standard; it is only in the binary
standard.

__readlink_chk

Name
__readlink_chk — display value of a symbolic link, with buffer overflow
checking

Synopsis
#include <unistd.h>
ssize_t __readlink_chk(const char * path, char * buf, size_t len,
size_t buflen);

Description
The interface __readlink_chk() shall function in the same way as the interface
readlink(), except that __readlink_chk() shall check for buffer overflow be-
fore computing a result. If an overflow is anticipated, the function shall abort
and the program calling it shall exit.
The parameter buflen specifies the size of the buffer buf. If len exceeds bu-
flen, the function shall abort, and the program calling it shall exit.
The __readlink_chk() function is not in the source standard; it is only in the binary
standard.
**__realpath_chk**

**Name**

__realpath_chk — return the canonicalized absolute pathname, with buffer overflow checking

**Synopsis**

```c
#include <stdlib.h>
char *__realpath_chk(const char *path, char *resolved_path, size_t resolved_len);
```

**Description**

The interface __realpath_chk() shall function in the same way as the interface realpath(), except that __realpath_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter resolved_len specifies the size of the string resolved_path. If resolved_len is less than PATH_MAX, then the function shall abort, and the program calling it shall exit.

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

**__recv_chk**

**Name**

__recv_chk — receive a message from a socket, with buffer overflow checking

**Synopsis**

```c
#include <sys/socket.h>
ssize_t __recv_chk(int fd, void *buf, size_t len, size_t buflen, int flag);
```

**Description**

The interface __recv_chk() shall function in the same way as the interface recv(), except that __recv_chk() shall check for buffer overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the buffer, parameter values, and so on.

The parameter buflen specifies the size of the buffer buf. If len exceeds buflen, the function shall abort, and the program calling it shall exit.

The __recv_chk() function is not in the source standard; it is only in the binary standard.
**__recvfrom_chk**

**Name**

__recvfrom_chk — receive a message from a socket, with buffer overflow checking

**Synopsis**

```c
#include <sys/socket.h>
ssize_t __recvfrom_chk(int fd, void * buf, size_t len, size_t buflen, int flag, struct sockaddr * from, socklen_t * fromlen);
```

**Description**

The interface __recvfrom_chk() shall function in the same way as the interface recvfrom(), except that __recvfrom_chk() shall check for buffer overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the buffer, parameter values, and so on.

The parameter buflen specifies the size of the buffer buf. If len exceeds buflen, the function shall abort, and the program calling it shall exit.

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

**__register_atfork**

**Name**

__register_atfork — alias for register_atfork

**Synopsis**

```c
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.

__snprintf_chk

Name

__snprintf_chk — convert formatted output, with buffer overflow checking

Synopsis

#include <stdio.h>
int __snprintf_chk(char * str, size_t maxlen, int flag, size_t strlen, const char * format);

Description

The interface __snprintf_chk() shall function in the same way as the interface
snprintf(), except that __snprintf_chk() shall check for buffer overflow be-
fore computing a result, depending on the value of the flag parameter. If an
overflow is anticipated, the function shall abort and the program calling it shall
exit.

In general, the higher the value of flag, the more security measures this inter-
face shall take in the form of checking the buffer, parameter values, and so on.

The parameter strlen specifies the size of the buffer str. If strlen is less than
maxlen, the function shall abort, and the program calling it shall exit.

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

Name

__sprintf_chk — convert formatted output, with stack checking

Synopsis

#include <stdio.h>
int __sprintf_chk(char * str, int flag, size_t strlen, const char * format);

Description

The interface __sprintf_chk() shall function in the same way as the interface sprintf(), except that __sprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

The parameter strlen specifies the size of the string str. If strlen is zero, the function shall abort, and the program calling it shall exit.

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

__stack_chk_fail

Name

__stack_chk_fail — terminate a function in case of stack overflow

Synopsis

#include <libc.h>
void __stack_chk_fail(void);

Description

The interface __stack_chk_fail() shall abort the function that called it with a message that a stack overflow has been detected. The program that called the function shall then exit.

Application Usage (informative)

The interface __stack_chk_fail() does not check for a stack overflow itself. It merely reports one when invoked.
__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.

__stpcpy_chk

Name
__stpcpy_chk — copy a string returning a pointer to its end, with buffer
overflow checking

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

Description
The interface __stpcpy_chk() shall function in the same way as the interface
stpcpy(), except that __stpcpy_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest.
The __stpcpy_chk() function is not in the source standard; it is only in the bi-
nary standard.
__stpncpy_chk

Name

__stpncpy_chk — copy a fixed-size string, returning a pointer to its end, with
buffer overflow checking

Synopsis

#include <libc.h>
char *__stpncpy_chk(char *dest, const char *src, size_t n, size_t destlen);

Description

The interface __stpncpy_chk() shall function in the same way as the interface
stpncpy(), except that __stpncpy_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest. If n
exceeds destlen, the function shall abort, and the program calling it shall exit.

The __stpncpy_chk() function is not in the source standard; it is only in the bi-
inary standard.

__strcat_chk

Name

__strcat_chk — concatenate two strings, with buffer overflow checking

Synopsis

#include <string.h>
char *__strcat_chk(char *dest, const char *src, size_t destlen);

Description

The interface __strcat_chk() shall function in the same way as the interface
strcat(), except that __strcat_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest.

The __strcat_chk() function is not in the source standard; it is only in the bi-
inary standard.
__strcpy_chk

Name
__strcpy_chk — copy a string, with buffer overflow checking

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

Description
The interface __strcpy_chk() shall function in the same way as the interface
strcpxy(), except that __strcpy_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest.

The __strcpy_chk() function is not in the source standard; it is only in the bi-
nary 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.
__strncat_chk

Name

__strncat_chk — concatenate two strings, with buffer overflow checking

Synopsis

#include <string.h>
char * __strncat_chk(char * s1, const char * s2, size_t n, size_t s1len);

Description

The interface __strncat_chk() shall function in the same way as the interface
strncat(), except that __strncat_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter s1len specifies the size of the object pointed to by s1.

The __strncat_chk() function is not in the source standard; it is only in the bi-
nary standard.

__strncpy_chk

Name

__strncpy_chk — copy a string, with buffer overflow checking

Synopsis

#include <string.h>
char * __strncpy_chk(char * s1, const char * s2, size_t n, size_t s1len);

Description

The interface __strncpy_chk() shall function in the same way as the interface
strncpy(), except that __strncpy_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter s1len specifies the size of the object pointed to by s1.

The __strncpy_chk() function is not in the source standard; it is only in the bi-
nary 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
strtol(__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 str-
told(__nptr, __endptr).
__strtold_internal() is not in the source standard; it is only in the binary
standard.
__strtoll_internal

Name
__strtoll_internal — underlying function for strtoll

Synopsis

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

Description
__group shall be 0 or the behavior of __strtoll_internal() is undefined.
__strtoll_internal(__nptr, __endptr, __base, 0) has the same specification as strtoll(__nptr, __endptr, __base).

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

__strtoul_internal

Name
__strtoul_internal — underlying function for strtoul

Synopsis

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

Description
__group shall be 0 or the behavior of __strtoul_internal() is undefined.
__strtoul_internal(__nptr, __endptr, __base, 0) has the same specification as strtoul(__nptr, __endptr, __base).

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

Name
__strtoull_internal — underlying function for strtoull

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

Description
__group shall be 0 or the behavior of __strtoull_internal() is undefined.
__strtoull_internal(__nptr, __endptr, __base, 0) has the same specification as strtoull(__nptr, __endptr, __base).
__strtoull_internal() is not in the source standard; it is only in the binary standard.

__swprintf_chk

Name
__swprintf_chk — convert formatted wide-character output, with stack checking

Synopsis
#include <wchar.h>
int __swprintf_chk(wchar_t * s, size_t n, int flag, size_t slen, const wchar_t * format);

Description
The interface __swprintf_chk() shall function in the same way as the interface swprintf(), except that __swprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.
The parameter slen specifies the size of the object pointed to by s. If slen is less than maxlen, the function shall abort and the program calling it shall exit.
In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.
The __swprintf_chk() function 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.

__syslog_chk

Name
__syslog_chk — send messages to the system logger, with stack checking

Synopsis
#include <syslog.h>
void __syslog_chk(int priority, int flag, const char * format);

Description
The interface __syslog_chk() shall function in the same way as the interface syslog(), except that __syslog_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

The __syslog_chk() function 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

— external variable containing timezone

Synopsis

long int __timezone;

Description

The external variable __timezone shall implement the timezone variable timezone as specified in ISO POSIX (2003). __timezone has the same specification as timezone.
__ttname_r_chk

Name

__ttname_r_chk — return name of a terminal, with buffer overflow checking (reentrant)

Synopsis

#include <unistd.h>
int __ttname_r_chk(int fd, char * buf, size_t buflen, size_t nreal);

Description

The interface __ttname_r_chk() shall function in the same way as the interface ttyname_r(), except that __ttname_r_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the object pointed to by buf. If buflen exceeds nreal, the function shall abort and the program calling it shall exit.

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

__tzname

Name

— external variable containing the timezone names

Synopsis

char *__tzname[2];

Description

The external variable __tzname shall implement the timezone name variable tzname as specified in ISO POSIX (2003) function tzset(). __tzname has the same specification as tzname.
___vfprintf_chk

Name
___vfprintf_chk — convert formatted output, with stack checking

Synopsis
#include <libc.h>
int ___vfprintf_chk(FILE * fp, int flag, const char * format, va_list ap);

Description
The interface ___vfprintf_chk() shall function in the same way as the interface
vfprintf(), except that ___vfprintf_chk() shall check for stack overflow be-
fore computing a result, depending on the value of the flag parameter. If an
overflow is anticipated, the function shall abort and the program calling it shall
exit.

In general, the higher the value of flag, the more security measures this inter-
face shall take in the form of checking the stack, parameter values, and so on.
The ___vfprintf_chk() function is not in the source standard; it is only in the
binary standard.

___vfprintf_chk

Name
___vfprintf_chk — convert formatted wide-character output, with stack
checking

Synopsis
#include <wchar.h>
int ___vfprintf_chk(FILE * fp, int flag, const wchar_t * format, va_list ap);

Description
The interface ___vfprintf_chk() shall function in the same way as the inter-
face vfprintf(), except that ___vfprintf_chk() shall check for stack over-
flow before computing a result, depending on the value of the flag parameter.
If an overflow is anticipated, the function shall abort and the program calling it shall
exit.

In general, the higher the value of flag, the more security measures this inter-
face shall take in the form of checking the stack, parameter values, and so on.
The ___vfprintf_chk() function is not in the source standard; it is only in the
binary standard.
**__vprintf_chk**

**Name**

__vprintf_chk — convert formatted output, with stack checking

**Synopsis**

```c
#include <stdio.h>
int __vprintf_chk(int flag, const char * format, va_list ap);
```

**Description**

The interface __vprintf_chk() shall function in the same way as the interface vprintf(), except that __vprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

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

---

**__vsnprintf_chk**

**Name**

__vsnprintf_chk — convert formatted output, with stack checking

**Synopsis**

```c
#include <stdio.h>
int __vsnprintf_chk(char * s, size_t maxlen, int flag, size_t slen, const char * format, va_list args);
```

**Description**

The interface __vsnprintf_chk() shall function in the same way as the interface vsnprintf(), except that __vsnprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

The parameter slen specifies the size of the object pointed to by s. If slen is less than maxlen, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

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

Name
__vsprintf_chk — convert formatted output, with stack checking

Synopsis
#include <stdio.h>
int __vsprintf_chk(char * s, int flag, size_t slen, const char * format, va_list args);

Description
The interface __vsprintf_chk() shall function in the same way as the interface vsprintf(), except that __vsprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

The parameter slen specifies the size of the object pointed to by s. If its value is zero, the function shall abort and the program calling it shall exit.

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

__vswprintf_chk

Name
__vswprintf_chk — convert formatted wide-character output, with stack checking

Synopsis
#include <wchar.h>
int __vswprintf_chk(wchar_t * s, size_t maxlen, int flag, size_t slen, const wchar_t * format, va_list args);

Description
The interface __vswprintf_chk() shall function in the same way as the interface vswprintf(), except that __vswprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

The parameter slen specifies the size of the object pointed to by s. If slen is less than maxlen, the function shall abort and the program calling it shall exit.

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

Name

__vsyslog_chk — send messages to the system logger, with stack checking

Synopsis

#include <syslog.h>
void __vsyslog_chk(int priority, int flag, const char * format, va_list ap);

Description

The interface __vsyslog_chk() shall function in the same way as the interface vsyslog(), except that __vsyslog_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

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

__vwprintf_chk

Name

__vwprintf_chk — convert formatted wide-character output, with stack checking

Synopsis

#include <wchar.h>
int __vwprintf_chk(int flag, const wchar_t * format, va_list ap);

Description

The interface __vwprintf_chk() shall function in the same way as the interface vwprintf(), except that __vwprintf_chk() shall check for stack overflow before computing a result, depending on the value of the flag parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

In general, the higher the value of flag, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

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

Name

__wcpcpy_chk — copy a wide-character string, returning a pointer to its end, with buffer overflow checking

Synopsis

#include <wchar.h>
wchar_t * __wcpcpy_chk(wchar_t * dest, const wchar_t * src, size_t destlen);

Description

The interface __wcpcpy_chk() shall function in the same way as the interface wcpcpy(), except that __wcpcpy_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest.

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

__wcpncpy_chk

Name

__wcpncpy_chk — copy a fixed-size string of wide characters, returning a pointer to its end, with buffer overflow checking

Synopsis

#include <wchar.h>
wchar_t * __wcpncpy_chk(wchar_t * dest, const wchar_t * src, size_t n, size_t destlen);

Description

The interface __wcpncpy_chk() shall function in the same way as the interface wcpncpy(), except that __wcpncpy_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest. If n exceeds destlen, the function shall abort and the program calling it shall exit.

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

Name
__wcrtomb_chk — convert a wide character to a multibyte sequence, with buffer overflow checking

Synopsis
#include <wchar.h>
size_t __wcrtomb_chk(char * s, wchar_t wchar, mbstate_t * ps, size_t buflen);

Description
The interface __wcrtomb_chk() shall function in the same way as the interface wcrtomb(), except that __wcrtomb_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the object pointed to by s. If it is less than MB_CUR_MAX, then the function shall abort and the program calling it shall exit.

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

__wcscat_chk

Name
__wcscat_chk — concatenate two wide-character strings, with buffer overflow checking

Synopsis
#include <wchar.h>
wchar_t * __wcscat_chk(wchar_t * dest, const wchar_t * src, size_t destlen);

Description
The interface __wcscat_chk() shall function in the same way as the interface wcscat(), except that __wcscat_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest.

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

**Name**

__wcscpy_chk — copy a wide-character string, with buffer overflow checking

**Synopsis**

```c
#include <wchar.h>
wchar_t *__wcscpy_chk(wchar_t * dest, const wchar_t * src, size_t n);
```

**Description**

The interface __wcscpy_chk() shall function in the same way as the interface wcsncpy(), except that __wcscpy_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

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

__wcsncat_chk

**Name**

__wcsncat_chk — concatenate two wide-character strings, with buffer overflow checking

**Synopsis**

```c
#include <wchar.h>
wchar_t *__wcsncat_chk(wchar_t * dest, const wchar_t * src, size_t n, size_t destlen);
```

**Description**

The interface __wcsncat_chk() shall function in the same way as the interface wcsncat(), except that __wcsncat_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest.

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

Name

__wcsncpy_chk — copy a fixed-size string of wide characters, with buffer
overflow checking

Synopsis

#include <wchar.h>
wchar_t * __wcsncpy_chk(wchar_t * dest, const wchar_t * src, size_t n, size_t destlen);

Description

The interface __wcsncpy_chk() shall function in the same way as the interface
wcsncpy(), except that __wcsncpy_chk() shall check for buffer overflow before
computing a result. If an overflow is anticipated, the function shall abort and
the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest. If len
exceeds destlen, the function shall abort and the program calling it shall exit.

The __wcsncpy_chk() function is not in the source standard; it is only in the bi-
nary standard.

__wcsnrtombs_chk

Name

__wcsnrtombs_chk — convert a wide-character string to a multibyte string,
with buffer overflow checking

Synopsis

#include <wchar.h>
size_t __wcsnrtombs_chk(char * dest, const wchar_t ** src, size_t nwc, size_t len, mbstate_t * ps, size_t destlen);

Description

The interface __wcsnrtombs_chk() shall function in the same way as the inter-
face wcsnrtombs(), except that __wcsnrtombs_chk() shall check for buffer
overflow before computing a result. If an overflow is anticipated, the function
shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest. If len
exceeds destlen, the function shall abort and the program calling it shall exit.

The __wcsnrtombs_chk() function is not in the source standard; it is only in the bi-
nary standard.
__wcsrtombs_chk

Name

__wcsrtombs_chk — convert a wide-character string to a multibyte string, with buffer overflow checking

Synopsis

#include <wchar.h>

size_t __wcsrtombs_chk(char * dest, const wchar_t * * src, size_t len, mbstate_t * ps, size_t destlen);

Description

The interface __wcsrtombs_chk() shall function in the same way as the interface wcsrtombs(), except that __wcsrtombs_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by dest. If len exceeds destlen, the function shall abort and the program calling it shall exit.

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

__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

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.

__wcstombs_chk

Name

__wcstombs_chk — convert a wide-character string to a multibyte string, with buffer overflow checking

Synopsis

#include <stdlib.h>
size_t __wcstombs_chk(char * dest, const wchar_t * src, size_t len, size_t destlen);

Description

The interface __wcstombs_chk() shall function in the same way as the interface wcstombs(), except that __wcstombs_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.
The parameter `destlen` specifies the size of the object pointed to by `dest`. If `len` exceeds `destlen`, the function shall abort and the program calling it shall exit.
The __wcstombs_chk() function is not in the source standard; it is only in the binary standard.
__wcstoul_internal

Name

__wcstoul_internal — underlying function for wcstoul

Synopsis

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

Description

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

__wcstoul_internal(nptr, endptr, base, 0)() 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.

__wctomb_chk

Name

__wctomb_chk — convert a wide character to a multibyte sequence, with buffer overflow checking

Synopsis

#include <stdlib.h>
int __wctomb_chk(char * s, wchar_t wchar, size_t buflen);

Description

The interface __wctomb_chk() shall function in the same way as the interface wctomb(), except that __wctomb_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter buflen specifies the size of the object pointed to by s. If it is less than MB_CUR_MAX, then the function shall abort and the program calling it shall exit.

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

Name

__wmemcpy_chk — copy an array of wide-characters, with buffer overflow checking

Synopsis

#include <wchar.h>

wchar_t * __wmemcpy_chk(wchar_t * s1, const wchar_t * s2, size_t n, size_t ns1);

Description

The interface __wmemcpy_chk() shall function in the same way as the interface wmemcpy(), except that __wmemcpy_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter ns1 specifies the size of the object pointed to by s1. If n exceeds ns1, the function shall abort and the program calling it shall exit.

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

__wmemmove_chk

Name

__wmemmove_chk — copy an array of wide-characters, with buffer overflow checking

Synopsis

#include <wchar.h>

wchar_t * __wmemmove_chk(wchar_t * s1, const wchar_t * s2, size_t n, size_t ns1);

Description

The interface __wmemmove_chk() shall function in the same way as the interface wmemmove(), except that __wmemmove_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter ns1 specifies the size of the object pointed to by s1. If n exceeds ns1, the function shall abort and the program calling it shall exit.

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

Name
__wmempcpy_chk — copy memory area, with buffer overflow checking

Synopsis
#include <wchar.h>
wchar_t * __wmempcpy_chk(wchar_t * s1, const wchar_t * s2, size_t n, size_t ns1);

Description
The interface __wmempcpy_chk() shall function in the same way as the interface wmempcpy(), except that __wmempcpy_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter ns1 specifies the size of the object pointed to by s1. If n exceeds ns1, the function shall abort and the program calling it shall exit.

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

__wmemset_chk

Name
__wmemset_chk — fill an array of wide-characters with a constant wide character, with buffer overflow checking

Synopsis
#include <wchar.h>
wchar_t * __wmemset_chk(wchar_t * s, wchar_t c, size_t n, size_t destlen);

Description
The interface __wmemset_chk() shall function in the same way as the interface wmemset(), except that __wmemset_chk() shall check for buffer overflow before computing a result. If an overflow is anticipated, the function shall abort and the program calling it shall exit.

The parameter destlen specifies the size of the object pointed to by s. If n exceeds destlen, the function shall abort and the program calling it shall exit.

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

**Name**

__wprintf_chk — convert formatted wide-character output, with stack checking

**Synopsis**

#include <wchar.h>

int __wprintf_chk(int flag, const wchar_t * format);

**Description**

The interface __wprintf_chk() shall function in the same way as the interface wprintf(), except that __wprintf_chk() shall check for stack overflow before computing a result, depending on the value of the *flag* parameter. If an overflow is anticipated, the function shall abort and the program calling it shall exit. In general, the higher the value of *flag*, the more security measures this interface shall take in the form of checking the stack, parameter values, and so on.

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

__xmknod

**Name**

__xmknod — make a special file

**Synopsis**

#include <sys/stat.h>

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

**Description**

The __xmknod() function shall implement the mknod() interface. The behavior of __xmknod() for values of *ver* other than _MKNOD_VER is undefined. See Data Definitions in the architecture specific part of this specification for the correct value of _MKNOD_VER.

__xmknod(_MKNOD_VER, path, mode, dev) shall behave as mknod(path, mode, dev) as specified by ISO POSIX (2003).

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.
__xmknodat

Name
__xmknodat — make a special file relative to a directory file descriptor

Synopsis

#include <sys/stat.h>
int __xmknodat(int ver, int dirfd, const char * path, mode_t path,
dev_t * dev);

Description
The __xmknodat() function shall implement the mknodat() function. The behavior of __xmknodat() for values of ver other than _MKNOD_VER is undefined. See Data Definitions in the architecture specific part of this specification for the correct value of _MKNOD_VER.

__xmknodat(_MKNOD_VER, dirfd, path, mode, dev) shall behave as mknodat(dirfd, path, mode, dev) as specified by POSIX 1003.1 2008.

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

Note: The mknodat() 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
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>
int __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 functions stat(), lstat(), and fstat() respectively.

The behavior of these functions for values of ver other than _STAT_VER is undefined. See Data Definitions in the architecture specific part of this specification for the correct value of _STAT_VER.

__xstat(_STAT_VER, path, stat_buf) shall implement stat(path, stat_buf) as specified by ISO POSIX (2003).

__lxstat(_STAT_VER, path, stat_buf) shall implement lstat(path, stat_buf) as specified by ISO POSIX (2003).

__fxstat(_STAT_VER, fildes, stat_buf) shall implement fstat(fildes, stat_buf) as specified by ISO POSIX (2003).

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

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

__xstat64

Name

__xstat64 — get File Status

Synopsis

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

**Description**

The functions __xstat64(), __lxstat64(), and __fxstat64() shall implement the functions stat64(), lstat64(), and fstat64() respectively.

The behavior of these functions for values of `ver` other than _STAT_VER is undefined. See Data Definitions in the architecture specific part of this specification for the correct value of _STAT_VER.

__xstat64(_STAT_VER, path, stat_buf) shall behave as stat64(path, stat_buf) as specified by Large File Support.

__lxstat64(_STAT_VER, path, stat_buf) shall behave as lstat64(path, stat_buf) as specified by Large File Support.

__fxstat64(_STAT_VER, fildes, stat_buf) shall behave as fstat64(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 retarding it by the time specified by the timeval delta. If delta is negative, the clock is slowed down by incrementing it more slowly than normal until the correction is complete. If delta is positive, a larger increment than normal is used. The skew used to perform the correction is generally a fraction of one percent. Thus, the time is always a monotonically increasing function. A time correction from an earlier call to adjtime() may not be finished when adjtime() is called again. If olddelta is non-NULL, the structure pointed to will contain, upon return, the number of microseconds still to be corrected from the earlier call.

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

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.
alphasort64

Name

alphasort64 — Comparison function for directory scanning (Large File Support)

Synopsis

#include <dirent.h>
int alphasort64(const struct dirent64 ** d1, const struct dirent64 ** d2);

Description

alphasort64() is a large-file version of the alphasort() function as defined in POSIX 1003.1 2008. If differs only in that the d1 and d2 parameters are of type dirent64 instead of type dirent.

asprintf

Name

asprintf — write formatted output to a dynamically allocated string

Synopsis

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

Description

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

Return Value

Refer to fprintf().

Errors

Refer to fprintf().
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**

```c
#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**

ENOOMEM

Insufficient memory available to allocate return value.

**See Also**
**bindresvport**

**Name**

`bindresvport` — bind socket to privileged IP port

**Synopsis**

```c
#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 an anonymous privileged IP port, that is, arbitrarily selected from the range 512 through 1023.

If the bind is successful and `sin` is not `NULL`, and the port number bound to is returned in the `sin_port` member of `sin`. Any caller-supplied value of `sin_port` is ignored.

If `sin` is `NULL`, the address family is taken to be `AF_INET` and an available privileged port is bound to. Since there is no `sockaddr_in` structure, the port number chosen cannot be returned. The `getsockname()` may be used to query for this information.

**Return Value**

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

**Errors**

`bindresvport()` may fail in the same way as `bind()` in ISO POSIX (2003). The following additional or differing failures may occur:

**EADDRINUSE**

All privileged ports are in use.

**EAFNOSUPPORT**

The specified address is not a valid address for the address family of the specified socket, or the address family is not supported.

**EPFNOSUPPORT**

The same meaning as EAFNOSUPPORT. Some older implementations may return this error instead.

**Note:** At this time, only `AF_INET` is supported. Applications should be prepared for either the EAFNOSUPPORT or EPFNOSUPPORT error to be indicated.
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 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 misadventurously selecting the wrong or non-existent 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, dnggettext, dcgettext, dcngettext, textdomain, bind_textdomain_codeset
cfmakeraw

Name
cfmakeraw — get and set terminal attributes

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

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

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

termios_p->c_oflag &= ~OPOST;

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

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

termios_p->c_cflag |= CS8;

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

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

Name

cfsetspeed — set terminal input and output data rate

Synopsis

#include <termios.h>
int cfsetspeed(struct termios *t, 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

clearerr_unlocked

Name

clearerr_unlocked — non-thread-safe clearerr

Description

clearerr_unlocked() is the same as clearerr(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().
daemon

Name

daemon — run in the background

Synopsis

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

Description

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

Return Value

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

dcgettext

Name

dcgettext — perform domain and category specific lookup in message catalog

Synopsis

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

Description

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

Return Value

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

Errors
dcgettext() shall not modify the errno global variable.

See Also
ggettext, dgettext, ngettext, dngettext, dcgettext, 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_MESSAGES 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, dngettext, dcgettext, dcngettext, textdomain, bind_textdomain, 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

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

See `dcngettext()` for more information.

**See Also**

`gettext`, `dgettext`, `ngettext`, `dcgettext`, `dcngettext`, `textdomain`, `bindtextdomain`, `bind_textdomain_codeset`

### drand48_r

**Name**

*drand48_r* — reentranently generate pseudorandom numbers in a uniform distribution

**Synopsis**

```c
#include <stdlib.h>

int

drand48_r(struct drand48_data * buffer, double * result);
```

**Description**

The interface `drand48_r()` shall function in the same way as the interface `drand48()`, except that `drand48_r()` shall use the data in `buffer` instead of the global random number generator state.

Before it is used, `buffer` must be initialized, for example, by calling `lcong48_r()`, `seed48_r()`, or `srand48_r()`, or by filling it with zeroes.
duplocale

Name
duplocale — provide new handle for selection of locale

Synopsis
#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.
epoll_create

Name

epoll_create — open an epoll file descriptor

Synopsis

#include <sys/epoll.h>
int epoll_create(int size);

Description

The epoll API, which consists of the interfaces epoll_create(), epoll_ctl(),
and epoll_wait(), shall support all file descriptors compatible with poll().
These interfaces shall be usable in either level-triggered or edge-triggered mode.
In level-triggered mode, epoll has similar semantics to poll(), and can be used
as a faster replacement for it. In edge-triggered mode, epoll shall only report
events for a file descriptor when changes occur on it.
The epoll_create() interface shall open an epoll file descriptor by allocating
an event backing store of approximately size size. The size parameter is a hint
to the kernel about how large the event storage should be, not a rigidly-defined
maximum size.

Return Value

On success, epoll_create() shall return the file descriptor, a non-negative integer that shall
be used for subsequent epoll calls. It should be closed with the close() function.
On failure, epoll_create() shall return -1 and set errno as follows.

Errors

EINVAL

The size parameter is not positive.

ENFILE

The maximum number of open files has been reached by the system.

ENOMEM

Not enough memory to create the kernel object.

See Also

close(), epoll_ctl(), epoll_wait(), poll().
epoll_ctl

Name

epoll_ctl — control an epoll file descriptor

Synopsis

#include <sys/epoll.h>
int epoll_ctl(int epfd, int op, int fd, struct epoll_event * event);

Description

The interface epoll_ctl() shall control an epoll file descriptor.

The parameter epfd shall specify the epoll file descriptor to control.

The parameter op shall specify the operation to perform on the specified target file descriptor.

The parameter fd shall specify the target file descriptor on which to perform the specified operation.

The parameter event shall specify the object associated with the target file descriptor. The events member of the event parameter is a bit set composed of the event types listed below.

Event types

EPOLLERR
An error condition occurred on the target file descriptor. It shall not be necessary to set this event in events; this interface shall always wait for it.

EPOLLET
This event shall set edge-triggered behavior for the target file descriptor. The default epoll behavior shall be level-triggered.

EPOLLHUP
A hang up occurred on the target file descriptor. It shall not be necessary to set this event in events; this interface shall always wait for it.

EPOLLIN
The file is accessible to read() operations.

EPOLLONESHOT
This event shall set one-shot behavior for the target file descriptor. After epoll_wait() retrieves an event, the file descriptor shall be disabled and epoll shall not report any other events. To reenable the file descriptor with a new event mask, the user should invoke epoll_ctl() with EPOLL_CTL_MOD in the op parameter.

EPOLLOUT
The file is accessible to write() operations.

EPOLLPRI
Urgent data exists for read() operations.

EPOLLRDHUP
A stream socket peer closed the connection, or else the peer shut down the writing half of the connection.

Values of the op parameter

EPOLL_CTL_ADD
Associate event with the file described by fd, and add fd to the epoll descriptor epfd.

EPOLL_CTL_DEL
Remove fd from epfd, and ignore event, which can be NULL.

EPOLL_CTL_MOD
Change the event event associated with fd.

Return Value
On success, epoll_ctl() shall return 0.
On failure, epoll_ctl() shall return -1 and set errno as follows.

Errors
EBADF
The parameter epfd or the parameter fd is an invalid file descriptor.

EEXIST
The parameter op was EPOLL_CTL_ADD, but the file descriptor fd is already in epfd.

EINVAL
The parameter epfd is invalid, or it is the same as fd, or the operation specified by the parameter op is unsupported.

ENOENT
The parameter op was EPOLL_CTL_MOD or EPOLL_CTL_DEL, but the file descriptor fd is not in epfd.

ENOMEM
Not enough memory for the operation specified by the parameter op.

EPERM
The file specified by fd does not support epoll.

See Also
close(), epoll_create(), epoll_wait(), poll().
epoll_wait

Name

epoll_wait — wait for I/O events on an epoll file descriptor

Synopsis

#include <sys/epoll.h>
int epoll_wait(int epfd, struct epoll_event * events, int maxevents, int timeout);

Description

The interface epoll_wait() shall wait for events on the epoll file descriptor specified by the parameter epfd.

Upon success, the output parameter events shall refer to an area of memory containing epoll_event structures available to the caller. The data members of these structures shall contain the data set by the user with the interface epoll_ctl(). The events members shall contain the event bit field that was returned.

The parameter maxevents shall specify the maximum number of events that epoll_wait() may return in the output parameter events. The value of this parameter should be greater than 0.

The parameter timeout shall specify the maximum number of milliseconds that epoll_wait() shall wait for events. If the value of this parameter is 0, then epoll_wait() shall return immediately, even if no events are available, in which case the return code shall be 0. If the value of timeout is -1, then epoll_wait() shall block until either a requested event occurs or the call is interrupted.

Return Value

On success, epoll_wait() shall return the number of file descriptors that are ready for the I/O that was requested, or else 0 if no descriptors became ready during timeout.

On failure, epoll_wait() shall return -1 and set errno as follows.

Errors

EBADF

The parameter epfd is not a valid file descriptor.

EFAULT

The area of memory referenced by the parameter events cannot be accessed with write permissions.

EINTR

The call was interrupted by a signal handler before the timeout expired or any requested event took place.

EINVAL


The parameter `epfd` is not a valid epoll file descriptor, or else the parameter `maxevents` is less than or equal to 0.

See Also

`close()`, `epoll_ctl()`, `epoll_create()`, `poll()`.

**erand48_r**

**Name**

`erand48_r` — reentrantly generate pseudorandom numbers in a uniform distribution

**Synopsis**

```c
#include <stdlib.h>
int erand48_r(unsigned short[3] xsubi, struct drand48_data * buffer, double * result);
```

**Description**

The interface `erand48_r()` shall function in the same way as the interface `erand48()`, except that `erand48_r()` shall use the data in `buffer` instead of the global random number generator state.

Before it is used, `buffer` must be initialized, for example, by calling `lcong48_r()`, `seed48_r()`, or `srand48_r()`, or by filling it with zeroes.
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>
void error(int exitstatus, int errnum, const char * format, ...);

Description

error() shall print a message to standard error.

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

1. the program name. If the application has provided a function named error_print_progname(), error() shall call this to supply the program name; otherwise, error() uses the content of the global variable program_name.

2. the colon and space characters, then the result of using the printf-style format and the optional arguments.

3. if errnum is nonzero, error() shall add the colon and space characters, then the result of strerror(errnum).

4. a newline.

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

See Also

err(), errx()
errx

Name
errx — display formatted error message and exit

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

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

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

Return Value
None.

Errors
None.

See Also
error(), err()
fcntl

Name
fcntl — file control

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

Implementation may set O_LARGEFILE

According to 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 c99 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.

Additional flags
In addition to the available values for cmd, as documented in ISO POSIX (2003), this specification permits the following constants.

F_GETSIG shall get the number of the signal to be sent when input or output can occur. If the value is 0, then SIGIO shall be sent. Otherwise, the value retrieved shall be the signal sent, and the signal handler can discover more information when installed with the SA_SIGINFO flag.

F_SETSIG shall set the number of the signal to be sent when input or output can occur. If the value is 0, then SIGIO shall be sent. Otherwise, the value set shall be the signal to be sent, and the signal handler can discover more information when installed with the SA_SIGINFO flag.

F_GETLK64 is analogous to the F_GETLK constant in ISO POSIX (2003), but shall provide a 64-bit interface on non-64-bit architectures. It is identical to F_GETLK on a 64-bit machine, but is provided in 64-bit environments for source code consistency among architectures.

F_SETLK64 is analogous to the F_SETLK constant in ISO POSIX (2003), but shall provide a 64-bit interface on non-64-bit architectures. It is identical to F_SETLK on a 64-bit machine, but is provided in 64-bit environments for source code consistency among architectures.

F_SETLKW64 is analogous to the F_SETLKW constant in ISO POSIX (2003), but provides a 64-bit interface on non-64-bit architectures. It is identical to F_SETLKW on a 64-bit machine, but is provided in 64-bit environments for source code consistency among architectures.
feof_unlocked

Name
feof_unlocked — non-thread-safe feof

Description
feof_unlocked() is the same as feof(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

ferror_unlocked

Name
ferror_unlocked — non-thread-safe ferror

Description
ferror_unlocked() is the same as ferror(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

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().

fgetc_unlocked

Name
fgetc_unlocked — non-thread-safe fgetc

Description
fgetc_unlocked() is the same as fgetc(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().
fgets_unlocked

Name
fgets_unlocked — non-thread-safe fgets

Description
fgets_unlocked() is the same as fgets(), 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().

fgetws_unlocked

Name
fgetws_unlocked — non-thread-safe fgetws

Description
fgetws_unlocked() is the same as fgetws(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

fileno_unlocked

Name
fileno_unlocked — non-thread-safe fileno

Description
fileno_unlocked() is the same as fileno(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().
flock

Name

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

Synopsis

int flock(int fd, int operation);

Description

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

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

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

LOCK_UN
    Unlock.

LOCK_NB
    Don't block when locking. May be specified (by 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.

ENOLCK
The implementation ran out of memory for allocating lock records.

**fputc_unlocked**

**Name**

fputc_unlocked — non-thread-safe fputc

**Description**

fputc_unlocked() is the same as fputc(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

**fputs_unlocked**

**Name**

fputs_unlocked — non-thread-safe fputs

**Description**

fputs_unlocked() is the same as fputs(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

**fputwc_unlocked**

**Name**

fputwc_unlocked — non-thread-safe fputwc

**Description**

fputwc_unlocked() is the same as fputwc(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

**fputws_unlocked**

**Name**

fputws_unlocked — non-thread-safe fputws

**Description**

fputws_unlocked() is the same as fputws(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().
fread_unlocked

Name
fread_unlocked — non-thread-safe fread

Description
fread_unlocked() is the same as fread(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

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 %[ 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 a large-file version of the fstatfs() function.

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().

fwrite_unlocked

Name
fwrite_unlocked — non-thread-safe fwrite

Description
fwrite_unlocked() is the same as fwrite(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().
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 qualifi-
ded domain name (for example, in DNS).

The LSB does not include other NIS functions, nor does it specify how NIS may af-
flect other database functions. No conforming application can make use of this infor-
mation 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 spec-
ification may remove this interface. Application developers should avoid using
this interface where possible.
getdtablesize

Name

getdtablesize — get file descriptor table size (DEPRECATED)

Synopsis

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

Description

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

Note: The getdtablesize() function is deprecated. Portable applications should call sysconf() with the _SC_OPEN_MAX option instead.

Return Value

The getdtablesize() function returns the current soft limit as if obtained by a call to sysconf() with the _SC_OPEN_MAX option.

Errors

No errors are defined.
getgrent_r

Name
getgrent_r — reentrantly get entry in group file

Synopsis
#include <grp.h>
int getgrent_r(struct group * gbuf, char * buf, size_t buflen,
               struct group ** gbufp);

Description
The reentrant interface getgrent_r() shall function in the same way as the inter-
face getgrent(), except that getgrent_r() shall return the group name,
group password, and group members in buffers provided by the caller, rather
than as a pointer to static storage.
The parameter gbuf contains the struct group that was read from the stream, if
any.
The parameter buf contains additional strings, if any.
The parameter buflen specifies the size of buf.
The parameter *gbufp returns a pointer to the struct group in *gbuf.

Return Value
On success, getgrent_r() shall return 0, and *gbufp shall contain a pointer to
the result.
On failure, *gbufp shall contain NULL, and getgrent_r() shall return an error
as follows.

Errors
ENOENT
No more group entries.
ERANGE
Not enough buffer space. Specify a larger buffer and try again.
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 supple-
mentary groups for the user specified by user. On entry, ngroups shall refer to
an integer containing the maximum number of elements in the groups array.
The group group shall also be included in the values returned in groups. It is
expected that group would be specified as the user’s primary group from the
password file (obtainable via getpwnam() or a similar function).

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 actu-
ally copied, and shall set the value referenced by ngroups to this value.

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

If the number of groups exceeds the input ngroups value, then as well as re-
turning -1, ngroups shall be set to the number of groups that would have been
placed in groups if it had been large enough.

Note: In such a case, the caller can use the information returned to make a further
getgrouplist() call with a correctly sized groups array.

If user does not refer to a valid user on the system, then the behavior of this
function is undefined.

Errors

None defined.

See Also

groups()
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 gethostbyname2() function shall return a pointer to a hostent
structure if the requested entry was found, and a null pointer otherwise.

On unsuccessful completion, gethostbyname2() shall set h_errno as for gethostbyname() in ISO POSIX (2003).

Errors

The gethostbyname2() shall set h_errno as for gethostbyname() in ISO
POSIX (2003).
gethostbyname2_r

**Name**

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

**Synopsis**

```c
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 getaddrinfo() instead.

The gethostbyname2_r() function shall search the network host database for an entry with name `name`. gethostbyname2_r() is a reentrant version of gethostbyname(). 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 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

int gethostbyname_r(const char * restrict name, struct hostent * restrict result_buf, char * restrict buf, size_t buflen, struct hostent ** restrict result, int * restrict h_errnop);

Description

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_errnop.

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_errnop 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.

**Return Value**

If the load average could not be obtained, -1 is returned. Otherwise, the number of samples actually retrieved is returned.

getopt

**Name**

getopt — parse command line options

**Synopsis**

```c
#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 ‘:’.
"?" 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.

goptopt_long

Name

goptopt_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

goptopt_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:
- no_argument (or 0) if the option does not take an argument,
- required_argument (or 1) if the option requires an argument, or
- optional_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
#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.
getprotobynamename_r

Name

getprotobynamename_r — retrieve information from the network protocol database by protocol name, reentrantly

Synopsis

#include <netdb.h>

int getprotobynamename_r(const char * name, struct protoent * result_buf, char * buf, size_t buflen, struct protoent ** result);

Description

The getprotobynamename_r() function is a reentrant version of the getprotobynamename() function.

The getprotobynamename_r() function shall search the network protocol database for an entry with the name name.

If a matching entry is found in the database, this function shall copy the relevant information to the application-supplied protoent 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.

The array buf shall contain the string fields referenced by the protoent structure that was returned. The parameter buflen shall specify the array's size. 1024 bytes should be enough for most uses.

Return Value

On success, the getprotobynamename_r() function shall return 0. If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small.
getprotobynumber_r

Name

getprotobynumber_r — retrieve information from the network protocol database by protocol number, reentrantly

Synopsis

#include <netdb.h>

int getprotobynumber_r(int proto, struct protoent * result_buf, char * buf, size_t buflen, struct protoent ** result);

Description

The getprotobynumber_r() function is a reentrant version of the getprotobynumber() function.

The getprotobynumber_r() function shall search the network protocol database for an entry with protocol number proto.

If a matching entry is found in the database, this function shall copy the relevant information to the application-supplied protoent 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.

The array buf shall contain the string fields referenced by the protoent structure that was returned. The parameter buflen shall specify the array’s size. 1024 bytes should be enough for most uses.

Return Value

On success, the getprotobynumber_r() function shall return 0. If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small.
getprotoent_r

Name

getprotoent_r — read the next entry of the protocol database, reentrantly

Synopsis

#include <netdb.h>

int getprotoent_r(struct protoent * result_buf, char * buf, size_t buflen, struct protoent ** result);

Description

The getprotoent_r() function is a reentrant version of the getprotoent() function.

The getprotoent_r() function shall search the network protocol database for the next entry.

If the next entry is found in the database, this function shall copy the relevant information to the application-supplied protoent structure referenced by result_buf, and return a pointer to this structure in *result. If no next entry is found, *result shall be set to a null pointer.

The array buf shall contain the string fields referenced by the protoent structure that was returned. The parameter buflen shall specify the array’s size. 1024 bytes should be enough for most uses.

Return Value

On success, the getprotoent_r() function shall return zero.

If the return value was ENOENT, there were no more entries in the database.

If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small.
**getpwent_r**

**Name**

getpwent_r — reentrantly get entry in passwd file

**Synopsis**

```c
#include <pwd.h>
int getpwent_r(struct passwd * pwbuf, char * buf, size_t buflen, 
struct passwd ** pwbufp);
```

**Description**

The reentrant interface getpwent_r() shall function in the same way as the interface getpwent(), except that getpwent_r() shall return the user name, user password, GECOS field, home directory, and shell program in buffers provided by the caller, rather than as a pointer to static storage.

The parameter `pwbuf` contains the struct passwd that was read from the stream, if any.

The parameter `buf` contains additional strings, if any.

The parameter `buflen` specifies the size of `buf`.

The parameter `*pwbufp` returns a pointer to the struct passwd in `*pwbuf`.

**Return Value**

On success, getpwent_r() shall return 0, and `*pwbufp` shall contain a pointer to the result.

On failure, `*pwbufp` shall contain NULL, and getpwent_r() shall return an error as follows.

**Errors**

ENOENT

No more password entries.

ERANGE

Not enough buffer space. Specify a larger buffer and try again.
getservbyname_r

Name

getservbyname_r — retrieve information from the network services database by service name, reentrantly

Synopsis

#include <netdb.h>
int getservbyname_r(const char * name, const char * proto, struct servent * result_buf, char * buf, size_t buflen, struct servent ** result);

Description

The getservbyname_r() function is a reentrant version of the getservbyname() function.

The getservbyname_r() function shall search the network services database for an entry with the name name. The proto parameter shall restrict the search to entries with the specified protocol. If proto is NULL, getservbyname_r() may return entries with any protocol.

If a matching entry is found in the database, this function shall copy the relevant information to the application-supplied servent 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.

The array buf shall contain the string fields referenced by the servent structure that was returned. The parameter buflen shall specify the array's size. 1024 bytes should be enough for most uses.

Return Value

On success, the getservbyname_r() function shall return zero. If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small.
getservbyport_r

Name

ggetservbyport_r — retrieve information from the network services database by service port, reentrantly

Synopsis

#include <netdb.h>
int getservbyport_r(int port, const char *proto, struct servent *result_buf, char *buf, size_t buflen, struct servent **result);

Description

The getservbyport_r() function is a reentrant version of the getservbyport() function.

The getservbyport_r() function shall search the network services database for an entry with the port port. The proto parameter shall restrict the search to entries with the specified protocol. If proto is NULL, getservbyport_r() may return entries with any protocol.

If a matching entry is found in the database, this function shall copy the relevant information to the application-supplied servent 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.

The array buf shall contain the string fields referenced by the servent structure that was returned. The parameter buflen shall specify the array's size. 1024 bytes should be enough for most uses.

Return Value

On success, the getservbyport_r() function shall return zero. If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small.
getservent_r

Name

getservent_r — read the next entry of the network services database, reentrantly

Synopsis

#include <netdb.h>
int getservent_r(struct servent * result_buf, char * buf, size_t buflen, struct servent ** result);

Description

The getservent_r() function is a reentrant version of the getservent() function.

The getservent_r() function shall search the network services database for the next entry.

If the next entry is found in the database, this function shall copy the relevant information to the application-supplied servent structure referenced by result_buf, and return a pointer to this structure in *result. If no next entry is found, *result shall be set to a null pointer.

The array buf shall contain the string fields referenced by the servent structure that was returned. The parameter buflen shall specify the array's size. 1024 bytes should be enough for most uses.

Return Value

On success, the getservent_r() function shall return 0.

If the return value was ENOENT, there were no more entries in the database.

If the return value was ERANGE, the size of the buffer buf, indicated by buflen, was too small.

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, dngettext, dcgettext, dcngettext, textdomain, bindtextdomain, bind_textdomain_codeset
getutent

Name
getutent — access user accounting database entries

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

Description
The getutent() function shall read the next entry from the user accounting database.

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

Errors
None defined.

getutent_r

Name
getutent_r — access user accounting database entries

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

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

Return Value
On success, getutent_r() shall return 0 and set the location referenced by result to a pointer to buffer. Otherwise, getutent_r() shall return -1 and set the location referenced by result to NULL.
getwc_unlocked

Name
getwc_unlocked — non-thread-safe getwc

Description
getwc_unlocked() is the same as getwc(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

getwchar_unlocked

Name
getwchar_unlocked — non-thread-safe getwchar

Description
getwchar_unlocked() is the same as getwchar(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

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
glob64() 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 gi_flags;
    void (*gl_closedir) (void *);
    struct dirent64 *(*gl.readdir64) (void *);
    void (*gl_opendir) (const char *);
    int (*gl_lstat) (const char *, struct stat *);
    int (*gl_stat) (const char *, struct stat *)
}
glob64_t;

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

Other members are defined as follows:

`gl_flags`
- reserved for internal use

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

`gl readdir64`
- pointer to a function capable of reading entries in a large directory

`gl opendir`
- pointer to a function capable of opening a large directory

`gl stat`
- pointer to a function capable of returning file status for a large file

`gl lstat`
- pointer to a function capable of returning file status information for a large file or symbolic link

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

**Return Value**

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

- `GLOB_NOSPACE`
  - out of memory

- `GLOB_ABORTED`
  - read error

- `GLOB_NOMATCH`
  - no match found
globfree64

Name

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

Synopsis

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

Description

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

globfree64() is a large-file version of the globfree() function defined in ISO
POSIX (2003).

hcreate_r

Name

hcreate_r — allocate space for a hash search table, reentrantly

Synopsis

#include <search.h>
int hcreate_r(size_t nel, struct hsearch_data * htab);

Description

The hcreate_r() function is a reentrant version of the hcreate() function.

hcreate_r() shall initialize the object referenced by htab with a hash table con-
taining at least nel elements. Unlike its non-reentrant equivalent, hcreate(),
the hcreate_r() function may work with more than one hash table.

The memory for the htab object may be dynamically allocated. It must be ini-
tialized with 0 before hcreate_r() is called.

Return Value

On success, hcreate_r() shall return a non-zero value.

On failure, hcreate_r() shall return 0. This usually happens because not
enough memory was available.
hdestroy_r

Name
hdestroy_r — dispose of a hash search table, reentrantly

Synopsis
#include <search.h>
void hdestroy_r(struct hsearch_data * htab);

Description
The hdestroy_r() function is a reentrant version of the hdestroy() function. hdestroy_r() frees the resources allocated by hcreate_r() for the object htab.

hsearch_r

Name
hsearch_r — search a hash table, reentrantly

Synopsis
#include <search.h>
int hsearch_r(ENTRY item, ACTION action, ENTRY * * retval, struct hsearch_data * htab);

Description
The hsearch_r() is a reentrant version of the hsearch() function, but instead of operating on a single global hash table, hsearch_r() operates on the table described by the object that htab references. This object can be initialized with the function hcreate_r(). Unlike the hsearch() function, hsearch_r() returns a pointer to the found entry in the variable referred to by retval, rather than directly.

Return Value
On success, hsearch_r() shall return a non-zero value.
On failure, hsearch_r() shall return 0 and set errno to an appropriate value.

Errors
ENOMEM
action was set to ENTER, but the table was full.

ESRCH
action was set to FIND, but no matching element was found in the table.
inet_aton

Name

inet_aton — Internet address manipulation routine

Synopsis

#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 Signifi-
cant Byte first.

initgroups

Name

initgroups — initialize the supplementary group access list

Synopsis

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

**Description**

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

**Return Value**

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

**Errors**

- **EPERM**
  The calling process does not have sufficient privileges.

- **ENOMEM**
  Insufficient memory to allocate group information structure.

**See Also**

- `setgroups()`

**initstate_r**

**Name**

`initstate_r` — reentrantly initialize a state array for random number generator functions

**Synopsis**

```
#include <stdlib.h>
int initstate_r(unsigned int seed, char *statebuf, size_t statelen, struct random_data *buffer);
```

**Description**

The interface `initstate_r()` shall function in the same way as the interface `initstate()`, except that `initstate_r()` shall use the data in `buffer` instead of the global random number generator state.
inotify_add_watch

Name

inotify_add_watch — add a watch to a watch list

Synopsis

#include <sys/inotify.h>
int inotify_add_watch(int fd, const char *path, uint32_t mask);

Description

inotify_add_watch() shall add a watch to, or modify an existing watch on, the watch list of the inotify instance specified by the file descriptor fd, for the file specified by path, to monitor the events specified by the bitmask mask. The caller must have read access to the file.

Return Value

On success, inotify_add_watch() shall return the unique, non-negative watch descriptor associated with the file path and the inotify instance specified by the file descriptor fd.

If path was already on the watch list, then inotify_add_watch() shall return the existing watch descriptor.

If path was not already on the watch list, then inotify_add_watch() shall allocate a new watch descriptor.

inotify_add_watch() shall not work recursively. Monitoring subdirectories of path shall require adding watches to them.

On failure, inotify_add_watch() shall return -1 and set errno to an appropriate value.

Errors

EACCESS

The caller does not have read access to path.

EBADF

The file descriptor fd is invalid.

EFAULT

path is outside of the address space accessible by the process.

EINVAL

mask contains no legal events, or fd is not a valid inotify file descriptor.

ENOMEM

There is not enough kernel memory available.

ENOSPC
The maximum number of watches has been created for this user, or the kernel cannot allocate a resource.

**Application Usage**

**Reading**

The function `read()` can be used to determine which inotify events have occurred. A blocking file descriptor will make `read()` block until at least one event has occurred.

If successful, `read()` will return at least one of the following `inotify_event` structures in a buffer:

```c
struct inotify_event {
    int     wd;
    uint32_t mask;
    uint32_t cookie;
    uint32_t len;
    char     path[];
};
```

`wd` is a watch descriptor that specifies the watch associated with the event. It is obtained from a previous invocation of `inotify_add_watch()`.

`mask` is a bit mask describing inotify events. See the section on masks below.

`cookie` is an integer associating related inotify events. The integer value is unique, and currently only enables the application to associate `IN_MOVE_FROM` and `IN_MOVE_TO` rename events.

`len` is a count of the bytes in `path`, including null bytes. This means that the total length of an `inotify_event` structure is

```
sizeof(inotify_event)+len
```
path is only returned when an event occurs for a file within a watched directory. This string is null-terminated, and it may contain more null bytes so that future reads will be aligned properly on an address boundary.

In kernels before 2.6.21, read() returns 0 when the buffer given to it is too small to return data about the next event. In subsequent kernels, it fails with the error EINVAL.

For a given file descriptor, the inotify events are returned in an ordered queue. Events on a file descriptor will always be returned in the correct order of occurrence. If two or more inotify events for a given file descriptor have identical values for all fields, then only one inotify_event will be returned to represent all of them.

The number of bytes that can be read from an inotify file descriptor can be determined by making a FIONREAD ioctl() call.

**Masks**

The *mask* argument of *inotify_add_watch()* and the mask field of the *inotify_event* structure are bit masks that specify inotify events. The bits in the list below can be set in the *mask* argument of *inotify_add_watch()* and returned in the mask field of *inotify_event*.

**IN_ACCESS**
- File was read.

**IN_ALL_EVENTS**
- Bit mask of all events in this list.

**IN_ATTRIB**
- File's metadata changed (including timestamps and permissions).

**IN_CLOSE**
- Same as
  - **IN_CLOSE_WRITE | IN_CLOSE_NOWRITE**
IN_CLOSE_WRITE
    File that was opened for writing was closed.

IN_CLOSE_NOWRITE
    File that was not opened for writing was closed.

IN_CREATE
    File or directory was created in a watched directory.

IN_DELETE
    File or directory was deleted in a watched directory.

IN_DELETE_SELF
    Watched file or directory was deleted.

IN_MODIFY
    File was changed.

IN_MOVE
    Same as
    IN_MOVED_FROM | IN_MOVED_TO
IN_MOVE_SELF
Watcher file or directory was moved

IN_MOVED_FROM
File was moved out of watched directory.

IN_MOVED_TO
File was moved into watched directory.

IN_OPEN
File was opened.

All of the events above, except for IN_DELETE_SELF and IN_MOVE_SELF, cause the name field of the inotify_event structure to contain the name of the file or directory being monitored.

The following bit is valid for inotify_add_watch() only.

IN_ONESHOT
Monitor path for an event, and then remove it from the watch list.

The following bits are valid for the inotify_event structure only.

IN_IGNORED
Watch was removed, either explicitly (via inotify_rm_watch()) or implicitly (file deletion or file system unmounting).

IN_ISDIR
Object being watched is a directory.

IN_Q_OVERFLOW
The event queue overflowed (wd is set to -1).

IN_UNMOUNT
File system of object being watched was unmounted.

Notes
It is possible to monitor file descriptors with the functions epoll(), poll(), and select().

When all of the file descriptors that point to an inotify instance have been closed, the instance and its associated resources and watches are freed by the kernel.

See Also

inotify_init(), inotify_rm_watch()
inotify_init

Name
inotify_init — instantiate inotify

Synopsis
#include <sys/inotify.h>
int inotify_init(void);

Description
inotify_init() shall create one instance of inotify.

Return Value
On success, inotify_init() shall return a file descriptor pointing to the new inotify instance.
On failure, inotify_init() shall return -1 and set errno to an appropriate value.

Errors
EMFILE
The maximum number of inotify instances has been created for this user.

ENFILE
The maximum number of file descriptors has been created on the system.

ENOMEM
There is not enough kernel memory available.

See Also
inotify_add_watch(), inotify_rm_watch()
inotify_rm_watch

Name
inotify_rm_watch — remove a watch from an inotify watch list

Synopsis
#include <sys/inotify.h>
int inotify_rm_watch(int fd, uint32_t wd);

Description
inotify_rm_watch() shall remove the watch associated with the watch descriptor wd from the watch list of the inotify instance associated with the file descriptor fd.
If a watch is removed, its watch descriptor shall generate the IN_IGNORED event.

Return Value
On success, inotify_rm_watch() shall return 0.
On failure, inotify_rm_watch() shall return -1 and set errno to an appropriate value.

Errors
EBADF
The file descriptor fd is invalid.
EINVAL
wd is invalid, or fd is not a valid inotify file descriptor.

See Also
inotify_add_watch(), inotify_init()
ioctl

Name
ioctl — control device

Synopsis
#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.
ENOTTY
  The specified request does not apply to the kind of object that fildes references.
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
#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. `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_dstaddr` field is set with the point-to-point address.

**SIOCGIFNAME**

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

**SIOCGIFNETMASK**

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

**SIOCGIFMTU**

Get the Maximum Transmission Unit (MTU) size 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_mtu` field is set with the MTU. Note: The range of valid values for MTU varies for an interface depending on the interface type.

**FIONREAD**

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

**Note:** Some implementations may also support the use of `FIONREAD` on other types of file descriptor. However, the LSB only specifies its behavior for a socket related file descriptor.

**Return Value**

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

**Errors**

**EBADF**

`sockfd` is not a valid descriptor.

**EFAULT**

`argp` references an inaccessible memory area.

**ENOTTY**

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

**EINVAL**

Either `request` or `argp` is invalid.
ENOTCONN
The operation is only defined on a connected socket, but the socket wasn't connected.

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:
TIOCWSIZE
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.
**jrand48_r**

**Name**

`jrand48_r` — reentrantly generate pseudorandom numbers in a uniform distribution

**Synopsis**

```c
#include <stdlib.h>
int jrand48_r(unsigned short[3] xsubi, struct drand48_data * buffer, long int * result);
```

**Description**

The interface `jrand48_r()` shall function in the same way as the interface `jrand48()`, except that `jrand48_r()` shall use the data in `buffer` instead of the global random number generator state.

Before it is used, `buffer` must be initialized, for example, by calling `lcong48_r()`, `seed48_r()`, or `srand48_r()`, or by filling it with zeroes.

**kill**

**Name**

`kill` — send a signal

**Synopsis**

```c
#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
lcong48_r

Name
lcong48_r — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis
#include <libc.h>
int lcong48_r(unsigned short[7] param, struct drand48_data * buffer);

Description
The interface lcong48_r() shall function in the same way as the interface lcong48(), except that lcong48_r() shall use the data in buffer instead of the global random number generator state.

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.
lrand48_r

Name
lrand48_r — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis
#include <stdlib.h>
int lrand48_r(struct drand48_data * buffer, long int * result);

Description
The interface lrand48_r() shall function in the same way as the interface lrand48(), except that lrand48_r() shall use the data in buffer instead of the global random number generator state.

Before it is used, buffer must be initialized, for example, by calling lcong48_r(), seed48_r(), or srand48_r(), or by filling it with zeroes.

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:
The conversion can stop for three reasons:

- An invalid multibyte sequence has been encountered. In this case src is left pointing to the invalid multibyte sequence, (size_t)(-1) is returned, and errno is set to EILSEQ.

- The nms limit forces a stop, or len non-L'\0' wide characters have been stored at dest. In this case, src is left pointing to the next multibyte sequence to be converted, and the number of wide characters written to dest is returned.

- The multibyte string has been completely converted, including the terminating '\0' (which has the side effect of bringing back ps to the initial state). In this case, src is set to NULL, and the number of wide characters written to dest, excluding the terminating L'\0' character, is returned.

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

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

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

Return Value

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

Notes

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

Passing NULL as ps is not multi-thread safe.

memmem

Name

memmem — locate bytes

Synopsis

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

Description

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

Return Value

If needle is found, memmem() returns a pointer to it. If needlelen is 0, memmem
returns haystack. If needle is not found in haystack, memmem() returns NULL.

Notes

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

memrchr

Name

memrchr — scan memory for a character

Synopsis

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

Description

The memrchr() function shall locate the last occurence of c (converted to an un-
signed 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

memcpy()
**mkstemp64**

**Name**

`mkstemp64` — create a unique temporary file (Large File Support)

**Synopsis**

```c
#include <stdio.h>
#include <stdlib.h>
int mkstemp64(char * template);
```

**Description**

`mkstemp64()` shall generates a unique temporary file name from `template`. The last six characters of `template` shall be `xxxxxx` and these are replaced with a string that makes the file name unique; the file is then created and an open file descriptor returned as described for `mkstemp()`.

`mkstemp64()` is a large-file version of the `mkstemp()` function as defined in ISO POSIX (2003). The only difference is that the temporary file is opened with `open64()` instead of with `open()`.

**Return Value**

On success, `mkstemp64()` returns the file descriptor of the temporary file. Otherwise `mkstemp64()` shall return -1 and set `errno` to indicate the error.

**Errors**

See `mkstemp()` for possible error values.

**mrand48_r**

**Name**

`mrand48_r` — reentrantly generate pseudorandom numbers in a uniform distribution

**Synopsis**

```c
#include <stdlib.h>
int mrand48_r(struct drand48_data * buffer, long int * result);
```

**Description**

The interface `mrand48_r()` shall function in the same way as the interface `mrand48()`, except that `mrand48_r()` shall use the data in `buffer` instead of the global random number generator state.

Before it is used, `buffer` must be initialized, for example, by calling `lcong48_r()`, `seed48_r()`, or `srand48_r()`, or by filling it with zeroes.
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
newlocale() 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 cate-
gories selected in category_mask. The category_mask value is a bitwise inclu-
sive 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

ggettext, dgettext, ngettext, dngettext, dcgettext, dcngettext, textdomain, bindtextdomain, bind_textdomain_codeset
**nrnd48_r**

**Name**

nrnd48_r — reentrantly generate pseudorandom numbers in a uniform distribution

**Synopsis**

```c
#include <stdlib.h>
int nrnd48_r(unsigned short[3] xsubi, struct drand48_data * buffer, long int * result);
```

**Description**

The interface nrnd48_r() shall function in the same way as the interface nrnd48(), except that nrnd48_r() shall use the data in buffer instead of the global random number generator state.

Before it is used, buffer must be initialized, for example, by calling lcong48_r(), seed48_r(), or srand48_r(), or by filling it with zeroes.

**openat64**

**Name**

openat64 — open a file relative to a directory file descriptor (Large File Support)

**Synopsis**

```c
#include <fcntl.h>
int openat64(int fd, const char *path, int oflag, ...);
```

**Description**

openat64() shall establish a connection between a file and a file descriptor. It shall be identical open64() except in the case where path specifies a relative path. In this case, the file to be opened shall be determined relative to the directory associated with the file descriptor fd instead of the current working directory.

openat64() is a large-file version of the openat() function as defined in POSIX 1003.1 2008. It differs from openat() in the same way that open64() differs from open(), that the open is done in large-file mode.

**Return Value**

On success, openat64() returns a new file descriptor. Otherwise openat64() shall return -1 and set errno to indicate the error.

**Errors**

See openat() for possible error values.
pmap_getport

Name

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

Synopsis

#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
#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 \([\text{program}, \text{version}, \text{protocol}]\) and \(\text{port}\) on the machine's RPC Bind service. The value of \(\text{protocol}\) 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 \([\text{prognum}, \text{versnum}, \text{*}]\) and \(\text{ports}\) 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.
putwc_unlocked

Name
putwc_unlocked — non-thread-safe putwc

Description
putwc_unlocked() is the same as putwc(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

putwchar_unlocked

Name
putwchar_unlocked — non-thread-safe putwchar

Description
putwchar_unlocked() is the same as putwchar(), except that it need not be thread-safe. That is, it may only be invoked in the ways which are legal for getc_unlocked().

random_r

Name
random_r — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis
#include <stdlib.h>
int random_r(struct random_data * buffer, int32_t * result);

Description
The interface random_r() shall function in the same way as the interface random(), except that random_r() shall use the data in buffer instead of the global random number generator state.

Before it is used, buffer must be initialized, for example, by calling lcong48_r(), seed48_r(), or srand48_r(), or by filling it with zeroes.
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.
scandir64

**Name**

scandir64 — scan a directory (Large File Support)

**Synopsis**

```c
#include <dirent.h>

int scandir64(const char *dir, const struct dirent64 **namelist, int (*sel) (const struct dirent64 *), int (*compar) (const struct dirent64 **, const struct dirent64 **));
```

**Description**

scandir64() is a large-file version of the scandir() function as defined in POSIX 1003.1 2008. If differs only in that the namelist and the paramters to the selection function sel and comparison function compar are of type dirent64 instead of type dirent.

**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 "%seconds" will have a different meaning on an LSB conforming system.
sched_getaffinity

Name

sched_getaffinity — retrieve the affinity mask of a process

Synopsis

#include <sched.h>
int sched_getaffinity(pid_t pid, unsigned int cpusetsize, cpu_set_t *mask);

Description

sched_getaffinity() shall retrieve the affinity mask of a process.

The parameter pid specifies the ID for the process. If pid is 0, then the calling
process is specified instead.

The parameter cpusetsize specifies the length of the data pointed to by mask,
in bytes. Normally, this parameter is specified as sizeof(cpu_set_t).

Return Value

On success, sched_getaffinity() shall return 0, and the structure pointed to
by mask shall contain the affinity mask of the specified process.

On failure, sched_getaffinity() shall return -1 and set errno as follows.

Errors

EFAULT
   Bad address.

EINVAL
   mask does not specify any processors that exist in the system, or
   cpusetsize is smaller than the kernel’s affinity mask.

ESRCH
   The specified process could not be found.

See Also

sched_setscheduler(), sched_setaffinity().
sched_setaffinity

Name
sched_setaffinity — set the CPU affinity mask for a process

Synopsis
#include <sched.h>
int sched_setaffinity(pid_t pid, unsigned int cpusetsize, cpu_set_t *mask);

Description
sched_setaffinity() shall set the CPU affinity mask for a process.
The parameter pid specifies the ID for the process. If pid is 0, then the calling
process is specified instead.
The parameter cpusetsize specifies the length of the data pointed to by mask,
in bytes. Normally, this parameter is specified as sizeof(cpu_set_t).
The parameter mask specifies the new value for the CPU affinity mask. The
structure pointed to by mask represents the set of CPUs on which the process
may run. If mask does not specify one of the CPUs on which the specified
process is currently running, then sched_setaffinity() shall migrate the
process to one of those CPUs.
Setting the mask on a multiprocessor system can improve performance. For ex-
ample, setting the mask for one process to specify a particular CPU, and then
setting the mask of all other processes to exclude the CPU, dedicates the CPU to
the process so that the process runs as fast as possible. This technique also pre-
vents loss of performance in case the process terminates on one CPU and starts
again on another, invalidating cache.

Return Value
On success, sched_setaffinity() shall return 0.
On failure, sched_setaffinity() shall return -1 and set errno as follows.

Errors
EFAULT
Bad address.
EINVAL
mask does not specify any processors that exist in the system, or
cpusetsize is smaller than the kernel's affinity mask.
EPERM
Insufficient privileges. The effective user ID of the process calling
sched_setaffinity() is not equal to the user ID or effective user ID of the
specified process, and the calling process does not have appropriate
privileges.
ESRCH
The specified process could not be found.

See Also

sched_setscheduler(), sched_getaffinity().

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.

seed48_r

Name

seed48_r — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis

#include <stdlib.h>
int seed48_r(unsigned short[3] seed16v, struct drand48_data * buffer);

Description

The interface seed48_r() shall function in the same way as the interface seed48(), except that seed48_r() shall use the data in buffer instead of the global random number generator state.
sendfile

Name

sendfile — transfer data between two file descriptors

Synopsis

#include <sys/sendfile.h>

ssize_t sendfile(int out_fd, int in_fd, off_t *offset, size_t count);

Description

The sendfile() function shall copy data between the file descriptor in_fd, which must not be a socket, and the file descriptor out_fd, which must be a socket. in_fd should be opened for reading, and out_fd should be opened for writing.

The offset parameter points to a variable set to the file offset at which sendfile() shall start reading from in_fd, unless it is NULL. On exit, this variable shall contain the offset of the byte immediately after the last byte read. sendfile() shall not change the current file offset of in_fd, unless it is NULL. In that case, sendfile() shall adjust the current file offset to show how many bytes were read.

The count parameter specifies how many bytes to copy.

Return Value

On success, sendfile() shall return the number of bytes written to out_fd.

On failure, sendfile() shall return -1 and set errno appropriately, as follows.

Errors

EAGAIN
Non-blocking I/O with O_NONBLOCK has been chosen, but the write would block.

EBADF
The input file is not open for reading, or the output file is not open for writing.

EFAULT
Bad address.

EINVAL
An mmap()-like operation is unavailable for in_fd, or file descriptor is locked or invalid.

EIO
There was an unspecified error while reading.

ENOMEM

There is not enough memory to read from \textit{in\_fd}.

\textbf{Notes}

\texttt{sendfile()} is usually faster than combining \texttt{read()} and \texttt{write()} calls, because it is part of the kernel. However, if it fails with \texttt{EINVAL}, falling back to \texttt{read()} and \texttt{write()} may be advisable.

It is advisable for performance reasons to use the \texttt{TCP\_CORK} option of the \texttt{tcp()} function when sending header data with file contents to a TCP socket. This minimizes the number of packets.

\textbf{See Also}

\texttt{mmap()}, \texttt{open()}, \texttt{socket()}, \texttt{splice()}.

\texttt{sendfile64}

\textbf{Name}

\texttt{sendfile64} — transfer data between two file descriptors (Large File Support)

\textbf{Synopsis}

\begin{verbatim}
#include <sys/sendfile.h>
ssize_t sendfile64(int out_fd, int in_fd, off64_t * offset, size_t count);
\end{verbatim}

\textbf{Description}

The \texttt{sendfile64()} function is a large-file version of the \texttt{sendfile()} function.

\texttt{setbuffer}

\textbf{Name}

\texttt{setbuffer} — stream buffering operation

\textbf{Synopsis}

\begin{verbatim}
#include <stdio.h>
void setbuffer(FILE * stream, char * buf, size_t size);
\end{verbatim}

\textbf{Description}

\texttt{setbuffer()} is an alias for the call to \texttt{setvbuf()}. It works the same, except that the size of the buffer in \texttt{setbuffer()} is up to the caller, rather than being determined by the default \texttt{BUFSIZ}. © 2008 Linux Foundation
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>


```c
#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/IEC 23360 Part 1:2008(E) 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 host name 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)`.

© 2008 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. `optval` 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. `optval` shall point to an integer which contains the new flag value.

**IP_ADD_MEMBERSHIP**

Join a multicast group. `optval` 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.

**IP_DROP_MEMBERSHIP**

© 2008 Linux Foundation
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).

---

### setstate_r

**Name**

`setstate_r` — reentrantly change the state array used by random number generator functions

**Synopsis**

```c
#include <stdlib.h>
int setstate_r(char * statebuf, struct random_data * buf);
```

**Description**

The interface `setstate_r()` shall function in the same way as the interface `setstate()`, except that `setstate_r()` shall use the data in `statebuf` instead of the global random number generator state.
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**

```c
#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.
srand48_r

Name
srand48_r — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis
#include <stdlib.h>
int srand48_r(long int seedval, struct drand48_data * buffer);

Description
The interface srand48_r() shall function in the same way as the interface srand48(), except that srand48_r() shall use the data in buffer instead of the global random number generator state.

srandom_r

Name
srandom_r — reentrantly set the seed for a new sequence of pseudorandom numbers

Synopsis
#include <stdlib.h>
int srandom_r(unsigned int seed, struct random_data * buffer);

Description
The interface srandom_r() shall function in the same way as the interface srandom(), except that srandom_r() shall use the data in buffer instead of the global random number generator state.
sscanf

Name
sscanf — 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.
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 a large-file version of the statfs() function.

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
```

© 2008 Linux Foundation

#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;
ISO/IEC 23360 Part 1:2008(E)

```c
  to = stpcpy (to, "foo");
  to = stpcpy (to, "bar");
  printf ("%s\n", buffer);
}
```

### stpncpy

**Name**

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

**Synopsis**

```c
#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**

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

**Description**

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

**Return Value**

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

Name

strerror_r — return string describing error number

Synopsis

#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 the error number errnum. 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
ENOOMEM
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.
strftime

Name

strftime — parse a time string

Description

The strftime() 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
```

© 2008 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.

### strt0q

#### Name

`strt0q` — 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.

`strtouq`

## Name

`strtouq` — convert a string to an unsigned long long

## Synopsis

```c
#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**

```c
#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 exec1p() 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

ENOOMEM

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**

```c
#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**

```c
#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**

```c
#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**

```c
#include <stdarg.h>
#include <err.h>
void verrx(int eval, const char * fmt, va_list args);
```

**Description**

The verrx() function 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 %[ 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.

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 %[ 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**

```c
#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.
vwscanf

Name

vwscanf — 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.

wait4

Name

wait4 — wait for process termination, BSD style

Synopsis

#include <sys/types.h>
#include <sys/resource.h>
#include <sys/wait.h>
pid_t wait4(pid_t pid, int * status, int options, struct rusage * rusage);

**Description**

`wait4()` suspends execution of the current process until a child (as specified by `pid`) has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child (as requested by `pid`) has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed.

The value of `pid` can be one of:

- `< -1`
  wait for any child process whose process group ID is equal to the absolute value of `pid`.

- `-1`
  wait for any child process; this is equivalent to calling `wait3()`.

- `0`
  wait for any child process whose process group ID is equal to that of the calling process.

- `> 0`
  wait for the child whose process ID is equal to the value of `pid`.

The value of `options` is a bitwise or of zero or more of the following constants:

- `WNOHANG`
  return immediately if no child is there to be waited for.

- `WUNTRACED`
  return for children that are stopped, and whose status has not been reported.

If `status` is not NULL, `wait4()` stores status information in the location `status`. This status can be evaluated with the following macros:

**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.

- `WEXITSTATUS(status)`
  evaluates to the least significant eight bits of the return code of the child that terminated, which may have been set as the argument to a call to `exit()` or as the argument for a return statement in the main program. This macro can only be evaluated if `WIFEXITED()` returned nonzero.

- `WIFSIGNALED(status)`
returns true if the child process exited because of a signal that was not caught.

WTERMSIG(status)
returns the number of the signal that caused the child process to terminate. This macro can only be evaluated if WIFSIGNALED() returned nonzero.

WIFSTOPPED(status)
returns true if the child process that caused the return is currently stopped; this is only possible if the call was done using WUNTRACED().

WSTOPSIG(status)
returns the number of the signal that caused the child to stop. This macro can only be evaluated if WIFSTOPPED() returned nonzero.

If rusage is not NULL, the struct rusage (as defined in sys/resource.h) that it points to will be filled with accounting information. See getrusage() 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.
warn

Name
warn — formatted error messages

Synopsis
#include <err.h>
void warn(const char *fmt, ...);

Description
The warn() function shall display a formatted error message on the standard error stream. The output shall consist of the last component of the program name, a colon character, and a space character. If fmt is non-NULL, it shall be used as a format string for the printf() family of functions, and the formatted message, a colon character, and a space are written to stderr. Finally, the error message string affiliated with the current value of the global variable errno shall be written to stderr, followed by a newline character.

Return Value
None.

Errors
None.

warnx

Name
warnx — formatted error messages

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

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

Return Value
None.

Errors
None.
wcpcpy

Name
wcpcpy — copy a wide character string, returning a pointer to its end

Synopsis

```c
#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

```c
#include <wchar.h>
wchar_t * wcpncpy(wchar_t * dest, const wchar_t * src, size_t n);
```

Description

wcpncpy() is the wide-character equivalent of stpcpy(). 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

#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 dif-
f erences (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 cur-
rent locale.
wcsdup

Name

wcsdup — duplicate a wide-character string

Synopsis

#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

ENOMEM

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`. 
wcstrnrtombs

Name

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

Synopsis

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

Description

wcsnrtombs() is like wcsrtombs(), except that the number of wide characters to be converted, starting at src, is limited to nwc.

If dest is not a NULL pointer, wcsnrtombs() converts at most nwc wide characters from the wide-character string src to a multibyte string starting at dest. At most len bytes are written to dest. The shift state ps is updated.

The conversion is effectively performed by repeatedly calling:

wcrtomb(dest, *src, ps)

as long as this call succeeds, and then incrementing dest by the number of bytes written and src by 1.

The conversion can stop for three reasons:

• A wide character has been encountered that cannot be represented as a multi-byte 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.

**wcstolq**

**Name**

wcstolq — convert wide string to long long int representation

**Synopsis**

```c
#include <wchar.h>
long long int wcstolq(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);
```

**Description**

The wcstolq() function shall convert the initial portion of the wide string nptr to long long int representation. It is identical to wcstoll().

**Return Value**

Refer to wcstoll().

**Errors**

Refer to wcstoll().
wcstouq

Name
wcstouq — convert wide string to unsigned long long int representation

Synopsis

#include <wchar.h>
unsigned long long 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 "%seconds" 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 13-37 defines the library name and shared object name for the libm library

<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
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 13-38, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-38 libm - Math Function Interfaces

<p>|               | __finitefp [LSB] | __finitefp [LSB] | acos [SUSv3]       |
| clog10f [LSB]| clog10f [LSB]   | clog10f [LSB]   | clog10f [LSB]      |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>drem</td>
<td>[LSB]</td>
</tr>
<tr>
<td>dremf</td>
<td>[LSB]</td>
</tr>
<tr>
<td>dreml</td>
<td>[LSB]</td>
</tr>
<tr>
<td>erf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>erfc</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>erfcf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>erfcl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>exp</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>exp10</td>
<td>[LSB]</td>
</tr>
<tr>
<td>exp10f</td>
<td>[LSB]</td>
</tr>
<tr>
<td>f10</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>exp11f</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fabs</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fabsf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fabsl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fdim</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fdimf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fdiml</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>feenableexceopt</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fegetenv</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fegetexceptflag</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fegetround</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fesetround</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fesetexceoptflag</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>finite</td>
<td>[LSB]</td>
</tr>
<tr>
<td>finitef</td>
<td>[LSB]</td>
</tr>
<tr>
<td>finitel</td>
<td>[LSB]</td>
</tr>
<tr>
<td>floor</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>floorf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>floorl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmax</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmaxf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmaxl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmin</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fminf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fminl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmod</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmodf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmodl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>frexp</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>frexpf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>frexpl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>gamma</td>
<td>[LSB]</td>
</tr>
<tr>
<td>gammaf</td>
<td>[LSB]</td>
</tr>
<tr>
<td>gammacl</td>
<td>[LSB]</td>
</tr>
<tr>
<td>hypot</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>hypotf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>hypotl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ilogb</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ilogbl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>j0</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>j1</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>j1f</td>
<td>[LSB]</td>
</tr>
<tr>
<td>j0f</td>
<td>[LSB]</td>
</tr>
<tr>
<td>j1f</td>
<td>[LSB]</td>
</tr>
<tr>
<td>jn</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>jnf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>jnl</td>
<td>[LSB]</td>
</tr>
<tr>
<td>ldexpf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ldexpl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lgamma</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lgammaf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lgammal</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lgammal_r</td>
<td>[LSB]</td>
</tr>
<tr>
<td>llrint</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>llrintf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>llrintl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>llround</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>llroundf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>llroundl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log10f</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log10f1</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log1pl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log2</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log2f</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>log2l</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>logb</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>logbf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>logbl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>logf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>logl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lrint</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lrintf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lrintl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lround</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lroundf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>lroundl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>matherr</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>mod</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>modff</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>modfl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nanf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nanl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nearbyint</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nearbyintf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nearbyintl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nextafter</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nextafterf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nextafterl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nexttoward</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nexttowardf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nexttowardl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pow</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>powf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pow10</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pow10f</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pow10l</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for Math specified in Table 13-39, 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 13-39 libm - Math Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>drem</th>
<th>dremf</th>
<th>dreml</th>
<th>finite</th>
<th>finitef</th>
<th>finitel</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic data interfaces for Math specified in Table 13-40, with the full mandatory functionality as described in the referenced underlying specification.

### Table 13-40 libm - Math Data Interfaces

<table>
<thead>
<tr>
<th>signgam</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[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

#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 double complex cacosh(double complex);
extern float complex cacoshf(float complex);
extern long double complex cacoshl(long double complex);
extern long double complex cacosl(long double complex);
extern double complex carg(double complex);
extern float complex cargf(float complex);
extern long double complex cargl(long double complex);
extern double complex casin(double complex);
extern float complex casinf(float 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 double complex catanh(double complex);
extern float complex catanhf(float complex);
extern long double complex catanhf(long double complex);
extern double complex ccomplex(double complex);
extern float complex ccomplexf(float complex);
extern long double complex ccomplexl(long double complex);
extern double complex ccos(double complex);
extern float complex ccoss(float complex);
extern double complex ccosh(double complex);
extern float complex ccoshf(float complex);
extern long double complex ccoshl(long double 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 cexp(long double complex);
extern float complex cimag(float complex);
extern long double complex cimagl(float complex);
extern double complex clog(double complex);
extern float complex clog10(double complex);
extern double complex clog10l(double complex);
extern float complex clog10f(float complex);
extern long double complex clog10l(long double complex);
extern long double complex clog10f(float complex);
extern long double complex clogl(long double 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 cprojl(double complex);
extern long double complex cprojl(long double complex);
extern long double complex creall(long double complex);
extern double complex csinl(double complex);
extern float complex csinf(float complex);
extern long double complex csinhl(long double complex);
extern long double complex csqrtl(long double complex);
extern long double complex csqrtl(long double complex);
extern long double complex ctanl(long double complex);
extern long double complex ctanl(long double complex);

13.7.2 fenv.h

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

13.7.3 math.h

#define DOMAIN 1
#define SING 2

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

#define isnormal(x) ((fpclassify (x) == FP_NORMAL) /* Return nonzero value if X is neither zero, subnormal, Inf, n */
#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 0.6366197723675813490
```
#define M_LN2   0.69314718055994530942
#define M_SQRT1_2       0.70710678118654752440
#define M_PI_4  0.78539816339744830962
#define M_2_SQRTPI      1.12837916709551257390
#define M_SQRT2 1.41421356237309504880
#define M_LOG2E 1.4426950408889634074
#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       /* errno set by math functions. */
#define MATH_ERREXCEPT  2       /* Exceptions raised by math functions. */

#define isunordered(u, v)       
  (__extension__({ __typeof__(u) __u = (u); __typeof__(v) __v = (v);fpclassify (__u) == FP_NAN || fpclassify (__v) == FP_NAN; })) /* Return nonzero value if arguments are unordered. */
#define islessgreater(x, y)     
  (__extension__({ __typeof__(x) __x = (x); __typeof__(y) __y = (y);!isunordered (__x, __y) && (__x < __y || __y < __x); })) /* Return nonzero value if either X is less than Y or Y is less */
#define isless(x,y)     
  (__extension__({ __typeof__(x) __x = (x); __typeof__(y) __y = (y);!isunordered (__x, __y) && __x < __y; })) /* Return nonzero value if X is less than Y. */
#define islessequal(x, y)       
  (__extension__({ __typeof__(x) __x = (x); __typeof__(y) __y = (y);!isunordered (__x, __y) && __x <= __y; })) /* Return nonzero value if X is less than or equal to Y. */
#define isgreater(x,y)  
  (__extension__({ __typeof__(x) __x = (x); __typeof__(y) __y = (y);!isunordered (__x, __y) && __x > __y; })) /* Return nonzero value if X is greater than Y. */
#define isgreaterequal(x,y)     
  (__extension__({ __typeof__(x) __x = (x); __typeof__(y) __y = (y);!isunordered (__x, __y) && __x >= __y; })) /* Return nonzero value if X is greater than or equal to Y. */

extern int __finite(double);
extern int __finitef(float);
extern int __finitel(long double);
extern int __fpclassify(double);
extern int __fpclassifyf(float);
extern int __fpclassifyl(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 double acos(double);
extern float acosf(float);
extern double acosh(double);
extern float acoshf(float);
extern double asinh(double);
extern float asinf(float);
```
extern float asinhf(float);
extern long double asinhl(long double);
extern long double asinl(long double);
extern double atan(double);
extern double atan2(double, double);
extern float atan2f(float, float);
extern long double atan2l(long double, long double);
extern float atanf(float);
extern double atanh(double);
extern float atanhf(float);
extern long double atanhl(long double);
extern long double atanl(long double);
extern double cbrt(double);
extern float cbrtf(float);
extern long double cbrtl(long double);
extern double ceil(double);
extern float ceilf(float);
extern long double ceill(long double);
extern double copysign(double, double);
extern float copysignf(float, float);
extern long double copysignl(long double, long double);
extern double cos(double);
extern float cosf(float);
extern double cosh(double);
extern float coshf(float);
extern long double coshl(long double);
extern long double cosl(long double);
extern double drem(double, double);
extern float dremf(float, float);
extern long double dreml(long double, long double);
extern double erf(double);
extern double erfc(double);
extern float erfcf(float);
extern long double erfcl(long double);
extern float erff(float);
extern long double erfl(long double);
extern double exp(double);
extern double exp10(double);
extern float exp10f(float);
extern double exp2(double);
extern float exp2f(float);
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 double fmod(double, double);
extern float fmodf(float, float);
extern long double fmodal(long double, long double);
extern double fmax(double, float);
extern float fmaxf(float, float);
extern long double fmaxl(long double, float);
extern double fmin(double, float);
extern float fminf(float, float);
extern long double fminl(long double, float);
extern double floor(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 float fminf(float, float);
extern long double fminl(long double, long double);
extern double fmod(double, double);
extern float fmodf(float, float);
extern long double fmodl(long double, long double);
extern double frexp(double, int *);
extern float frexpf(float, int *);
extern long double frexpl(long double, int *);
extern double gamma(double);
extern float gammaf(float);
extern long double gammal(long double);
extern double hypot(double, double);
extern float hypotf(float, float);
extern long double hypotl(long double, long double);
extern int ilogb(double);
extern int ilogbf(float);
extern int ilogbl(long double);
extern double j0(double);
extern float j0f(float);
extern long double j0l(long double);
extern double j1(double);
extern float j1f(float);
extern long double j1l(long double);
extern double jn(int, double);
extern float jnf(int, float);
extern long double jnl(int, long double);
extern double ldexp(double, int);
extern float ldexpf(float, int);
extern long double ldexpl(long double, int);
extern double lgamma(double);
extern double lgamma_r(double, int *);
extern float lgammaf(float);
extern float lgammaf_r(float, int *);
extern long double lgammal(long double);
extern long double lgammal_r(long double, int *);
extern long long int llrint(double);
extern long long int llrintf(float);
extern long long int llrintl(long double);
extern long int llround(double);
extern long int llroundf(float);
extern long int llroundl(long double);
extern double modf(double, double *);
extern float modff(float, float *);
extern long double modfl(long double, 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 double nextafter(double, double);
extern float nextafterf(float, float);
extern long double nextafterl(long double, long double);
extern double nexttoward(double, long double);
extern float nexttowardf(float, long double);
extern long double nexttowardl(long double, long double);
extern double pow(double, double);
extern double pow10(double);
extern float pow10f(float);
extern long double pow10l(long double);
extern double 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 double remannderf(float, float);
extern float remainderl(float, float);
extern double rint(double);
extern float rintf(float);
extern long double rintl(long double);
extern double round(double);
extern float roundf(float);
extern long double roundl(float);
extern double scalb(double, double);
extern float scalbf(float, float);
extern long double scalbl(long double, long double);
extern double scalbln(double, long int);
extern float scalbf(float, float);
extern long double scalbl(long double, long int);
extern double scalbn(double, int);
extern float scalbnf(float, int);
extern long double scalbln1(long double, long int);
extern double scalbn(double, int);
extern float scalbnf(float, int);
extern long double scalbln1(long double, long int);
extern int signgam;
extern double significand(double);
extern float significandf(float);
extern long double significandl(long double);
extern double sin(double);
extern float sinf(float);
extern double sinh(double);
extern float sinhf(float);
extern double sinh1(double);
extern long double sinl(long double);
extern double sqrt(double);
extern float sqrtf(float);
extern double sqrt1(double);
extern double tan(double);
extern float tanf(float);
extern double tanh(double);
extern float tanhf(float);
extern double tanh1(double);
extern long double tanl(long double);
extern double tgamma(double);
extern float tgammaf(float);
extern long double tgammal(long double);
extern double trunc(double);
extern float truncf(float);
extern long double truncf(long double);
13 Base Libraries

ISO/IEC 23360 Part 1:2008(E)

extern double y0(double);
extern float y0f(float);
extern long double y0l(long double);
extern double y1(double);
extern float y1f(float);
extern long double y1l(long double);
extern double yn(int, double);
extern float ynf(int, float);
extern long double yn1(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**

```c
#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**

```c
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**

```c
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

clg10l — Logarithm of a Complex Number

Synopsis

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

Description
The drem() 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 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**

```c
#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](https://www.chiappp.com/iec60559/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**

```c
#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 \(10^x\).

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 \(10^x\).

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 \(10^x\).

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 fegetexcept() function returns the set of all currently enabled exceptions.

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
**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**

```c
#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()`
Name

gammaf — log gamma function (DEPRECATED)

Synopsis

#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

#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()
j0f

Name
j0f — Bessel functions

Synopsis
#include <math.h>
float j0f(float x);

Description
The j0f() function is identical to j0(), except that the argument x and the return value is a float.

Returns
See j0().

See Also
j0(), j0l(), j1(), j1f(), j1l(), jn(), jnf(), jnl(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), yn1()
**j1f**

**Name**

j1f — Bessel functions

**Synopsis**

```c
#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()`, `jn1()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

---

**j1l**

**Name**

j1l — Bessel functions

**Synopsis**

```c
#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()`, `jn1()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`
**jnf**

**Name**

jnf — Bessel functions

**Synopsis**

```c
#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**

```c
#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(), jnf(), y0(), y0f(), y0l(), y1(), y1f(), y1l(), yn(), ynf(), ynl()
**lgamma_r**

**Name**

`lgamma_r` — log gamma functions

**Synopsis**

```c
#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`
lgammal_r

Name
lgammal_r — log gamma functions

Synopsis
#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 lgamma(). 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 lgamma() and signgam.

See Also
lgamma(), lgammal_r(), lgammaf_r(), signgam

pow10

Name
pow10 — Base-10 power function

Synopsis
#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 rised 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()
scalbf

Name
scalbf — load exponent of radix-independent floating point number

Synopsis
#include <math.h>
float scalbf(float x, double exp);

Description
The scalbf() function is identical to scalb(), except that the argument x and the return value is of type float.

Returns
See scalb().

scalbl

Name
scalbl — load exponent of radix-independent floating point number

Synopsis
#include <math.h>
long double scalbl(long double x, double exp);

Description
The scalbl() function is identical to scalb(), except that the argument x and the return value is of type long double.

Returns
See scalb().
significand

Name

significand — floating point mantissa

Synopsis

#include <math.h>
double significand(double x);

Description

The significand() function shall return the mantissa of x, 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.

This function is equivalent to scalb(x, (double)-ilogb(x)).

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**

```c
#include <math.h>
float significandf(float x);
```

**Description**

The `significandf()` function shall return the mantissa of `x`, `sig` such that \( x \equiv sig \times 2^n \) 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, `significandf()` shall return the mantissa of `x` in the range \( 1 \leq sig < 2 \).

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 \( x \), \( \text{sig} \) such that \( x \equiv \text{sig} \times 2^n \) 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.

This function is equivalent to `scalb(x, (double)-ilogb(x))`.

**Returns**

Upon successful completion, significandl() shall return the mantissa of \( x \) in the range \( 1 \leq \text{sig} < 2 \).

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 \( \text{sin} \), and the cosine in the location referenced by \( \text{cos} \).

**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(), jn1(), y0(), 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(), jn1(), y0(), y0f(), 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(), y1f(), 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(), y1l(), yn(), ynf(), ynl()
**ynf**

**Name**
ynf — Bessel functions

**Synopsis**

```c
#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()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`

**ynl**

**Name**

ynl — Bessel functions

**Synopsis**

```c
#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()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`

### 13.9 Interfaces for libpthread

Table 13-41 defines the library name and shared object name for the libpthread library

© 2008 Linux Foundation
The behavior of the interfaces in this library is specified by the following specifications:

[LF] Large File Support
[LS] This Specification
[SUSv4] POSIX 1003.1 2008

### 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 13-42, with the full mandatory functionality as described in the referenced underlying specification.

#### Table 13-42 libpthread - Realtime Threads Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>SUSv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getinheritsched</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_attr_getschedpolicy</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_attr_getscope</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_attr_setschedpolicy</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_attr_setscope</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_getschedparam</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_setschedparam</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_setschedprio</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_mutexattr_getprioceiling</td>
<td>GLIBC_2.4</td>
<td>SUSv4</td>
</tr>
<tr>
<td>pthread_mutexattr_setprioceiling</td>
<td>GLIBC_2.4</td>
<td>SUSv4</td>
</tr>
<tr>
<td>pthread_mutexattr_getprotocol</td>
<td>GLIBC_2.4</td>
<td>SUSv4</td>
</tr>
<tr>
<td>pthread_mutexattr_setprotocol</td>
<td>GLIBC_2.4</td>
<td>SUSv4</td>
</tr>
<tr>
<td>pthread_getsockopt</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_setsockopt</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_getcancelstate</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_setcancelstate</td>
<td>SUSv3</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 13-43, with the full mandatory functionality as described in the referenced underlying specification.

#### Table 13-43 libpthread - Advanced Realtime Threads Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>SUSv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_barrier_destroy</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_barrier_init</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_barrier_wait</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_barrierattr_getpshared</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_barrierattr_setpshared</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_barrierattr_gettpshared</td>
<td>GLIBC_2.3.3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_barrierattr_settpshared</td>
<td>GLIBC_2.3.3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pthread_getcpuclockid</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
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 13-44, with the full mandatory functionality as described in the referenced underlying specification.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_lock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_trylock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_unlock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getdetachstate</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getguardsize</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getstack</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getstackaddr</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setdetachstate</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setguardsize</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setstack</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setstackaddr</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cancel</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_broadcast</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_signal</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_timedwait</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_wait</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_getpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_gettype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_setpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_settype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_create</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_detach</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_equal</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_exit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getconcurrency</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getspecific</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_join</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_key_create</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_key_delete</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_key_delete</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_lock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_trylock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_unlock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_getpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_gettype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_setpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_settype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_once</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_trylock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_unlock</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

Table 13-44 libpthread - Posix Threads Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_cleanp_pop</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pthread_cleanp_push</td>
<td>[LSB]</td>
</tr>
<tr>
<td>pthread_attr_getdetachstate</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getguardsize</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getstack</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getstackaddr</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setdetachstate</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setguardsize</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setschedparam</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setstack</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setstackaddr</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cancel</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_broadcast</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_signal</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_timedwait</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_wait</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_getpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_gettype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_setpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_settype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_create</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_detach</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_equal</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_exit</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getconcurrency</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_getspecific</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_join</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_key_create</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_key_delete</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_lock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_trylock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutex_unlock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_getpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_gettype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_setpshared</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_mutexattr_settype</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_once</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_destroy</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_init</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_trylock</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>pthread_rwlock_unlock</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic deprecated functions for Posix Threads specified in Table 13-45, 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 13-45 libpthread - Posix Threads Deprecated Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>[SUSv3]</th>
<th>[SUSv3]</th>
<th>[SUSv3]</th>
<th>[SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_get_stackaddr</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pthread_attr_set_stackaddr</td>
<td></td>
<td></td>
<td></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 13-46, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-46 libpthread - Thread aware versions of libc interfaces Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>[LFS]</th>
<th>[SUSv3]</th>
<th>[LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>lseek64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>open64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pread</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pwrite</td>
<td>[SUSv3]</td>
<td></td>
<td>[LSB]</td>
</tr>
<tr>
<td>pwrite64</td>
<td>[SUSv3]</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 __SIZEOF_PTHREAD_CONDATTR_T 4
#define __SIZEOF_PTHREAD_MUTEXATTR_T 4
#define __SIZEOF_PTHREAD_COND_T 48
#define __SIZEOF_PTHREAD_RWLOCKATTR_T 8
#define pthread_cleanup_push(routine, arg)       \
   {struct _pthread_cleanup_buffer _buffer;
    _pthread_cleanup_push(&_buffer,(routine),(arg));
#define  pthread_cleanup_pop(execute)
    _pthread_cleanup_pop(&_buffer,(execute));}
#define PTHREAD_COND_INITIALIZER        { { 0, 0, 0, 0, 0, (void *) 0, 0, 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 volatile int pthread_spinlock_t;
typedef union {
   char __size[__SIZEOF_PTHREAD_BARRIERATTR_T];
   int __align;
} pthread_barrierattr_t;
typedef unsigned long int pthread_t;

typedef union {
   struct __pthread_mutex_s __data;
   char __size[__SIZEOF_PTHREAD_MUTEX_T];
   long int __align;
} pthread_mutex_t;

typedef union {
   char __size[__SIZEOF_PTHREAD_MUTEXATTR_T];
   int __align;
} pthread_mutexattr_t;

typedef union {
   char __size[__SIZEOF_PTHREAD_ATTR_T];
   long int __align;
} pthread_attr_t;
```
typedef union {
    struct {
        int __lock;
        unsigned int __futex;
        unsigned long long int __total_seq;
        unsigned long long int __wakeup_seq;
        unsigned long long int __woken_seq;
        void *__mutex;
        unsigned int __nwaiters;
        unsigned int __broadcast_seq;
    } __data;
    char __size[__SIZEOF_PTHREAD_COND_T];
    long long int __align;
} pthread_cond_t;

typedef union {
    char __size[__SIZEOF_PTHREAD_CONDATTR_T];
    int __align;
} pthread_condattr_t;

typedef union {
    char __size[__SIZEOF_PTHREAD_RWLOCKATTR_T];
    long int __align;
} 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 __register_atfork(void (*)(void), void (*)(void),
    void (*)(void), void *);
extern void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *
    , int);
extern void _pthread_cleanup_push(struct _pthread_cleanup_buffer *
    ,
    void (*)(void *), void *);
extern int pthread_attr_destroy(pthread_attr_t * __attr);
extern int pthread_attr_getdetachstate(const pthread_attr_t * __attr,
    int *__detachstate);
extern int pthread_attr_getguardsize(const pthread_attr_t * __attr,
    size_t *__guardsize);
extern int pthread_attr_getinheritsched(const pthread_attr_t * __attr,
    int *__inherit);
extern int pthread_attr_getschedparam(const pthread_attr_t * __attr,
    struct sched_param *__param);
extern int pthread_attr_getschedpolicy(const pthread_attr_t * __attr,
    int *__policy);
extern int pthread_attr_getscope(const pthread_attr_t * __attr,
    int *__scope);
extern int pthread_attr_getstack(const pthread_attr_t * __attr,
    void **__stackaddr, size_t *
__stacksize);

extern int pthread_attr_getstackaddr(const pthread_attr_t * __attr,
void **__stackaddr);

extern int pthread_attr_getstacksize(const pthread_attr_t * __attr,
size_t *__stacksize);

extern int pthread_attr_init(pthread_attr_t * __attr);

extern int pthread_attr_setdetachstate(pthread_attr_t * __attr,
int __detachstate);

extern int pthread_attr_setguardsize(pthread_attr_t * __attr,
size_t __guardsize);

extern int pthread_attr_setinheritsched(pthread_attr_t * __attr,
int __inherit);

extern int pthread_attr_setscope(pthread_attr_t * __attr,
int __scope);

extern int pthread_attr_setstack(pthread_attr_t * __attr,
void *__stackaddr, size_t __stacksize);

extern int pthread_attr_setstackaddr(pthread_attr_t * __attr,
void **__stackaddr);

extern int pthread_attr_setstacksize(pthread_attr_t * __attr,
size_t __stacksize);

extern int pthread_barrier_destroy(pthread_barrier_t * __barrier);

extern int pthread_barrier_init(pthread_barrier_t * __barrier,
const pthread_barrierattr_t * __attr,
unsigned int __count);

extern int pthread_barrier_wait(pthread_barrier_t * __barrier);

extern int pthread_barrierattr_destroy(pthread_barrierattr_t * __attr);

extern int pthread_barrierattr_getpshared(const pthread_barrierattr_t * __attr,
int *__pshared);

extern int pthread_barrierattr_init(pthread_barrierattr_t * __attr);

extern int pthread_barrierattr_setpshared(pthread_barrierattr_t * __attr,
int __pshared);

extern int pthread_cancel(pthread_t __th);

extern int pthread_cond_broadcast(pthread_cond_t * __cond);

extern int pthread_cond_destroy(pthread_cond_t * __cond);

extern int pthread_cond_init(pthread_cond_t * __cond,
const pthread_condattr_t * __cond_attr);

extern int pthread_cond_signal(pthread_cond_t * __cond);

extern int pthread_cond_timedwait(pthread_cond_t * __cond,
pthread_mutex_t * __mutex,
const struct timespec *__abstime);

extern int pthread_cond_wait(pthread_cond_t * __cond,
pthread_mutex_t * __mutex);

extern int pthread_condattr_destroy(pthread_condattr_t * __attr);

extern int pthread_condattr_getpshared(const pthread_condattr_t * __attr,
int *__pshared);

extern int pthread_condattr_init(pthread_condattr_t * __attr);

extern int pthread_condattr_setpshared(pthread_condattr_t * __attr,
int __pshared);

extern int pthread_create(pthread_t * __newthread,
    const pthread_attr_t * __attr,
    void *(*__start_routine) (void *), void
    *__arg);

extern int pthread_detach(pthread_t __th);
extern int pthread_equal(pthread_t __thread1, pthread_t
    __thread2);
extern void pthread_exit(void *__retval);
extern int pthread_getpct concurrency(void);
extern int pthread_setconcurrency(pthread_t __thread_id,
clockid_t *__clock_id);
extern int pthread_equal(pthread_t __thread1, pthread_t
    __thread2);
extern int pthread_getconcurrency(void);

extern int pthread_getconcurrency(void);
extern int pthread_getpct concurrency(void);
extern int pthread_setconcurrency(pthread_t __thread_id,
clockid_t *__clock_id);
}

extern int pthread_equal(pthread_t __thread1, pthread_t
    __thread2);
extern void pthread_exit(void *__retval);
extern int pthread_getpct concurrency(void);
extern int pthread_setconcurrency(pthread_t __thread_id,
clockid_t *__clock_id);

extern int pthread_getspecific(pthread_key_t __key);
extern int pthread_getspecific(pthread_key_t __key);
extern int pthread_getspecific(pthread_key_t __key);

extern int pthread_mutex_destroy(pthread_mutex_t * __mutex);
extern int pthread_mutex_unlock(pthread_mutex_t * __mutex);
extern int pthread_mutex_trylock(pthread_mutex_t * __mutex);
extern int pthread_mutex_lock(pthread_mutex_t * __mutex);

extern int pthread_mutex_attr_destroy(pthread_mutexattr_t *
    __attr);
extern int pthread_mutexattr_destroy(pthread_mutexattr_t *
    __attr);
extern int pthread_mutexattr_destroy(pthread_mutexattr_t *
    __attr);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceiling);

extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
    __attr, int
    *__prioceilin
extern int pthread_once(pthread_once_t * __once_control, 
    void (*__init_routine) (void));
extern int pthread_rwlock_destroy(pthread_rwlock_t * __rwlock);
extern int pthread_rwlock_init(pthread_rwlock_t * __rwlock,
    const pthread_rwlockattr_t * __attr);
extern int pthread_rwlock_rdlock(pthread_rwlock_t * __rwlock);
extern int pthread_rwlock_timedrdlock(pthread_rwlock_t * __rwlock,
    const struct timespec *__abstime);
extern int pthread_rwlock_timedwrlock(pthread_rwlock_t * __rwlock,
    const struct timespec * __abstime);
extern int pthread_rwlock_tryrdlock(pthread_rwlock_t * __rwlock);
extern int pthread_rwlock_trywrlock(pthread_rwlock_t * __rwlock);
extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t * __attr);
extern int pthread_rwlockattr_getpshared(const pthread_rwlockattr_t * __attr, int * __pshared);
extern int pthread_rwlockattr_init(pthread_rwlockattr_t * __attr);
extern int pthread_rwlockattr_setpshared(pthread_rwlockattr_t * __attr,
    int __pshared);
extern pthread_t pthread_self(void);
extern int pthread_setcancelstate(int __state, int *__oldstate);
extern int pthread_setcanceltype(int __type, int *__oldtype);
extern int pthread_setconcurrency(int __level);
extern int pthread_setschedparam(pthread_t __target_thread, int __policy,
    const struct sched_param *__param);
extern int pthread_setschedprio(pthread_t __target_thread, int __prio);
extern int pthread_setspecific(pthread_key_t __key, const void *__pointer);
extern int pthread_spin_destroy(pthread_spinlock_t * __lock);
extern int pthread_spin_init(pthread_spinlock_t * __lock, int __pshared);
extern int pthread_spin_lock(pthread_spinlock_t * __lock);
extern int pthread_spin_trylock(pthread_spinlock_t * __lock);
extern int pthread_spin_unlock(pthread_spinlock_t * __lock);
extern void pthread_testcancel(void);

13.10.2 semaphore.h

typedef union {
    char __size[__SIZEOF_SEM_T];
    long int __align;
} sem_t;

#define SEM_FAILED ((sem_t*)0)
#define SEM_VALUE_MAX ((int)((~0u)>>1))

extern int sem_close(sem_t * __sem);
extern int sem_destroy(sem_t * __sem);
extern int sem_getvalue(sem_t * __sem, int *__sval);
extern int sem_init(sem_t * __sem, int __pshared, unsigned int

© 2008 Linux Foundation
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

#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.
pread64

Name

pread64 — read from a file (Large File Support)

Synopsis

#include <unistd.h>
ssize_t pread64(int fd, void * buf, size_t count, off64_t offset);

Description

pread64() shall read count bytes into buf from the file associated with the open file descriptor fd, at the position specified by offset, without changing the file position.

pread64() is a large-file version of the pread() function as defined in ISO POSIX (2003). It differs from pread() in that the offset parameter is an off64_t instead of an off_t.

Return Value

On success, pread64() shall return the number of bytes actually read. Otherwise pread64() shall return -1 and set errno to indicate the error.

Errors

See pread() for possible error values.
pwrite64

Name

pwrite64 — write on a file (Large File Support)

Synopsis

#include <unistd.h>
ssize_t pwrite64(int fd, const void *buf, size_t count, off64_t offset);

Description

pwrite64() shall write count bytes from buf to the file associated with the
open file descriptor fd, at the position specified by offset, without changing
the file position.

pwrite64() is a large-file version of the pwrite() function as defined in ISO
POSIX (2003). It differs from pwrite() in that the offset parameter is an
off64_t instead of an off_t

Return Value

On success, pwrite64() shall return the number of bytes actually written. Other-
wise pwrite() shall return -1 and set errno to indicate the error.

Errors

See pwrite() for possible error values.

13.12 Interfaces for libgcc_s

Table 13-47 defines the library name and shared object name for the libgcc_s li-
brary

<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 specifi-
cation.

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

typedef unsigned int _Unwind_Ptr __attribute__((__mode__(__pointer__)));
typedef unsigned int _Unwind_Word __attribute__((__mode__(__word__)));
typedef unsigned int _Unwind_Exception_Class __attribute__((__mode__(__DI__)));

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) (int _Unwind_Reason_Code,
                                                struct _Unwind_Exception *);

struct _Unwind_Exception {
    _Unwind_Exception_Class exception_class;
    _Unwind_Exception_Cleanup_Fn exception_cleanup;
    _Unwind_Word private_1;
    _Unwind_Word private_2;
} __attribute__((__aligned__));

#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 13-48 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 specifi-
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 13-49, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-49 libdl - Dynamic Loader Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>dladdr</td>
<td>[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dlclose</td>
<td></td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dlerror</td>
<td></td>
<td></td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>dlopen</td>
<td></td>
<td></td>
<td></td>
<td>[LSB]</td>
</tr>
<tr>
<td>dlsym</td>
<td>[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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_DEFAULT    ((void *) 0L)
#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 *__address, Dl_info * __info);
extern int dlclose(void *__handle);
extern char *dlerror(void);
extern void *dlopen(const char *__file, int __mode);
extern void *dlsym(void *__handle, const char *__name);
```
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

```c
#include <dlfcn.h>

typedef struct {
    const char *dli_fname;
    void   *dli_fbase;
    const char *dli_sname;
    void   *dli_saddr;
} dl_addr_t;
```

} Dl_info;
int dladdr(const void * addr, Dl_info * dlip);

Description

The dladdr() function shall query the dynamic linker for information about the shared object containing the address addr. The information shall be returned in the user supplied data structure referenced by dlip.

The structure shall contain at least the following members:

dli_fname
   The pathname of the shared object containing the address

dli_fbase
   The base address at which the shared object is mapped into the address space of the calling process.

dli_sname
   The name of the nearest runtime symbol with value less than or equal to addr. Where possible, the symbol name shall be returned as it would appear in C source code.

   If no symbol with a suitable value is found, both this field and dli_saddr shall be set to NULL.

dli_saddr
   The address of the symbol returned in dli_sname. 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**

```c
#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.

**RTLD_NEXT, RTLD_DEFAULT Required**

The values RTLD_NEXT and RTLD_DEFAULT, described as reserved for future use in ISO POSIX (2003), are required, with behavior as described in ISO POSIX (2003).

13.17 Interfaces for librt

Table 13-50 defines the library name and shared object name for the librt library

**Table 13-50 librt Definition**

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

© 2008 Linux Foundation
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 13-51, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-51 librt - Shared Memory Objects Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>shm_open</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>shm_unlink</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

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 13-52, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-52 librt - Clock Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock_gettime</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>clock_getres</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>clock_getcpucloc</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>clock_nanosleep</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>clock_settime</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

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 13-53, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-53 librt - Timers Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>timer_create</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>timer_delete</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>timer_gettime</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>timer_getoverrun</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>timer_settime</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

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 13-54, with the full mandatory functionality as described in the referenced underlying specification.
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 mqqueue.h

```c
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 __mqdes);
extern int mq_getattr(mqd_t __mqdes, struct mq_attr *__mqstat);
extern int mq_notify(mqd_t __mqdes, const struct sigevent *__notification);
extern mqd_t mq_open(const char *__name, int __oflag, ...);
extern ssize_t mq_receive(mqd_t __mqdes, char *__msg_ptr, size_t __msg_len,
                         unsigned int *__msg_prio);
extern int mq_send(mqd_t __mqdes, const char *__msg_ptr, size_t __msg_len,
                 unsigned int *__msg_prio);
extern int mq_setattr(mqd_t __mqdes, const struct mq_attr *__mqstat,
                    struct mq_attr *__omqstat);
extern ssize_t mq_timedreceive(mqd_t __mqdes, char *__msg_ptr,
                             size_t __msg_len, unsigned int *__msg_prio,
                             const struct timespec *__abs_timeout);
```

© 2008 Linux Foundation
extern int mq_timedsend(mqd_t __mqdes, const char *__msg_ptr, size_t __msg_len, unsigned int __msg_prio, const struct timespec *__abs_timeout);
extern int mq_unlink(const char *__name);

13.19 Interfaces for libcrypt

Table 13-55 defines the library name and shared object name for the libcrypt library

<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 13-56, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-56 libcrypt - Encryption Function Interfaces


13.20 Interfaces for libpam

Table 13-57 defines the library name and shared object name for the libpam library

<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 13-58, with the full mandatory functionality as described in the referenced underlying specification.

Table 13-58 libpam - Pluggable Authentication API Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pam_acct_mgmt</td>
<td>Accounting management</td>
</tr>
<tr>
<td>pam_authenticate</td>
<td>Authentication</td>
</tr>
<tr>
<td>pam_chauthtok</td>
<td>Change authentication</td>
</tr>
<tr>
<td>pam_close_session</td>
<td>Close session</td>
</tr>
<tr>
<td>pam_end</td>
<td>End session</td>
</tr>
<tr>
<td>pam_fail_delay</td>
<td>Fail delay</td>
</tr>
<tr>
<td>pam_get_item</td>
<td>Get item</td>
</tr>
<tr>
<td>pam_getenv</td>
<td>Get environment</td>
</tr>
<tr>
<td>pam_getenvlist</td>
<td>Get environment list</td>
</tr>
<tr>
<td>pam_open_session</td>
<td>Open session</td>
</tr>
<tr>
<td>pam_putenv</td>
<td>Put environment</td>
</tr>
<tr>
<td>pam_set_item</td>
<td>Set item</td>
</tr>
<tr>
<td>pam_strerror</td>
<td>Error string</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; /* currently un-used, zero expected */
};
struct pam_conv {
  int (*conv) (int num_msg, const struct pam_message * *msg,
               struct pam_response * *resp, void *appdata_ptr);
  void *appdata_ptr;
};

© 2008 Linux Foundation
```c
#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       /* The service name */
#define PAM_USER        2       /* The user name */
#define PAM_TTY 3               /* The tty name */
#define PAM_RHOST       4       /* The remote host name */
#define PAM_CONV        5       /* The pam_conv structure */
#define PAM_RUSER       8       /* The remote user name */
#define PAM_USER_PROMPT 9       /* the prompt for getting a username */
#define PAM_SUCCESS     0       /* Successful function return */
#define PAM_OPEN_ERR    1       /* dlopen() failure */
#define PAM_USER_UNKNOWN        10      /* User not known to the underlying authenticaiton module */
#define PAM_MAXTRIES 11       /* An authentication service has maintained a retry count which */
#define PAM_NEW_AUTHTOK_REQD    12      /* New authentication token required */
#define PAM_ACCT_EXPIRED        13      /* User account has expired */
#define PAM_SESSION_ERR 14      /* Can not make/remove an entry for the specified session */
#define PAM_CRED_UNAVAIL        15      /* Underlying authentication service can not retrieve user cred */
#define PAM_CRED_EXPIRED        16      /* User credentials expired */
#define PAM_CRED_ERR    17      /* Failure setting user credentials */
#define PAM_CONV_ERR    19      /* Conversation error */
#define PAM_SYMBOL_ERR  2       /* Symbol not found */
#define PAM_AUTHTOK_ERR  20       /* Authentication token manipulation error */
#define PAM_AUTHTOK_RECOVER_ERR  21       /* Authentication information cannot be recovered */
#define PAM_AUTHTOK_LOCK_BUSY  22      /* Authentication token aging disabled */
#define PAM_TRY_AGAIN 24      /* Preliminary check by password service */
#define PAM_ABORT       26      /* Critical error (?module fail now request) */
#define PAM_AUTHTOK_EXPIRED     27      /* user's authentication token has expired */
#define PAM_BAD_ITEM 29       /* Bad item passed to pam_*_item() */
#define PAM_SERVICE_ERR 3       /* Error in service module */
#define PAM_SYSTEM_ERR 4       /* System error */
#define PAM_BUF_ERR      5       /* Memory buffer error */
#define PAM_PERM_DENIED 6       /* Permission denied */
#define PAM_AUTH_ERR    7       /* Authentication failure */
#define PAM_CRED_INSUFFICIENT   8       /* Can not access authentication data due to insufficient crede */
#define PAM_AUTHINFO_UNAVAIL     9        /* Underlying authentication service can not retrieve authentic */
#endif
```
credentials */
#define PAM_REFRESH_CRED 0x0010U /* Extend lifetime of user credentials */
#define PAM_CHANGE_EXPIRED_AUTHTOK 0x0020U /* Extend lifetime of user credentials */
#define PAM_SILENT 0x8000U /* Authentication service should not generate any messages */

extern int pam_acct_mgmt(pam_handle_t *, int);
extern int pam_authenticate(pam_handle_t *, int);
extern int pam_chauthtok(pam_handle_t *, int);
extern int pam_close_session(pam_handle_t *, int);
extern int pam_end(pam_handle_t *, int);
extern int pam_fail_delay(pam_handle_t *, unsigned int);
extern int pam_get_item(const pam_handle_t *, int, const void **);
extern const char *pam_getenv(pam_handle_t *, const char *);
extern char **pam_getenvlist(pam_handle_t *);
extern int pam_open_session(pam_handle_t *, int);
extern int pam_putenv(pam_handle_t *, const char *);
extern int pam_set_item(pam_handle_t *, int, const void *);
extern int pam_setcred(pam_handle_t *, int);
extern int pam_start(const char *, const char *, const struct pam_conv *,
pam_handle_t * *);
extern const char *pam_strerror(pam_handle_t *, int);

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_AUHTHTOK_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().
pam_authenticate

Name

pam_authenticate — authenticate the user

Synopsis

#include <security/pam_appl.h>
int pam_authenticate(pam_handle_t * pnh, int flags);

Description

pam_authenticate() serves as an interface to the authentication mechanisms of the loaded modules.

flags is an optional parameter that may be specified by the following value:

PAM_DISALLOW_NULL_AUTHTOK

Instruct the authentication modules to return PAM_AUTH_ERR if the user does not have a registered authorization token.

Additionally, the value of flags may be logically or'd with PAM_SILENT.

The process may need to be privileged in order to successfully call this function.

Return Value

PAM_SUCCESS

Success.

PAM_AUTH_ERR

User was not authenticated or process did not have sufficient privileges to perform authentication.

PAM_CRED_INSUFFICIENT

Application does not have sufficient credentials to authenticate the user.

PAM_AUTHINFO_UNAVAIL

Modules were not able to access the authentication information. This might be due to a network or hardware failure, etc.

PAM_USER_UNKNOWN

Supplied username is not known to the authentication service.

PAM_MAXTRIES

One or more authentication modules has reached its limit of tries authenticating the user. Do not try again.

PAM_ABORT

One or more authentication modules failed to load.

Note: Errors may be translated to text with pam_strerror().
pam_chauthtok

**Name**
pam_chauthtok — change the authentication token for a given user

**Synopsis**
#include <security/pam_appl.h>
int pam_chauthtok(pam_handle_t * pamh, const int flags);

**Description**
pam_chauthtok() is used to change the authentication token for a given user as indicated by the state associated with the handle pamh.

*flags* is an optional parameter that may be specified by the following value:

PAM_CHANGE_EXPIRED_AUTHTOK

User's authentication token should only be changed if it has expired.

Additionally, the value of *flags* may be logically or'd with PAM_SILENT.

**RETURN VALUE**
PAM_SUCCESS
Success.
PAM_AUTHTOK_ERR
A module was unable to obtain the new authentication token.
PAM_AUTHTOK_RECOVER_ERR
A module was unable to obtain the old authentication token.
PAM_AUTHTOK_LOCK_BUSY
One or more modules were unable to change the authentication token since it is currently locked.
PAM_AUTHTOK_DISABLE_AGING
Authentication token aging has been disabled for at least one of the modules.
PAM_PERM_DENIED
Permission denied.
PAM_TRY_AGAIN
Not all modules were in a position to update the authentication token(s). In such a case, none of the user's authentication tokens are updated.
PAM_USER_UNKNOWN
User is not known to the authentication token changing service.

**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**

```c
#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**

```c
#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

```
#include <security/pam_appl.h>
int pam_get_item(const pam_handle_t * pamh, int item_type, const void ** item);
```

Description
pam_get_item() obtains the value of the indicated item_type. The possible values of item_type are the same as listed for pam_set_item().

On success, item contains a pointer to the value of the corresponding item. Note that this is a pointer to the actual data and should not be free()’d or over-written.

Return Value

PAM_SUCCESS
Success.

PAM_PERM_DENIED
Application passed a NULL pointer for item.

PAM_BAD_ITEM
Application attempted to get an undefined item.

Note: Errors may be translated to text with pam_strerror().
pam_getenv

Name

pam_getenv — get a PAM environment variable

Synopsis

#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 environ-
ment 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 * pam);

Description

pam_getenvlist() returns a pointer to the complete PAM environment. This
pointer points to an array of pointers to NULL-terminated strings and must be ter-
minated by a NULL pointer. Each string has the form "name=value".
The PAM library module allocates memory for the returned value and the asso-
ciated strings. The calling application is responsible for freeing this memory.

Return Value

pam_getenvlist() returns an array of string pointers containing the PAM en-
vironment. 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

```c
#include <security/pam_appl.h>
extern int pam_setcred(pam_handle_t * pamh, int flags);
```

Description

pam_setcred() sets the module-specific credentials of the user. It is usually called after the user has been authenticated, after the account management function has been called and after a session has been opened for the user.

`flags` maybe specified from among the following values:

- **PAM_ESTABLISH_CRED**
  - set credentials for the authentication service
- **PAM_DELETE_CRED**
  - delete credentials associated with the authentication service
- **PAM_REINITIALIZE_CRED**
  - reinitialize the user credentials
- **PAM_REFRESH_CRED**
  - extend lifetime of the user credentials

Additionally, the value of `flags` may be logically or'd with **PAM_SILENT**.

Return Value

**PAM_SUCCESS**

Success.

**PAM_CRED_UNAVAIL**

Module cannot retrieve the user's credentials.

**PAM_CRED_EXPIRED**

User's credentials have expired.

**PAM_USER_UNKNOWN**

User is not known to an authentication module.

**PAM_CRED_ERR**

Module was unable to set the credentials of the user.

**Note:** Errors may be translated to text with `pam_strerror()`. 
pam_start

Name

pam_start — initialize the PAM library

Synopsis

#include <security/pam_appl.h>
int pam_start(const char * service_name, const char * user, const struct pam_conv * pam_conversation, pam_handle_t ** pamh);

Description

pam_start() is used to initialize the PAM library. It must be called prior to any other usage of the PAM library. On success, *pamh becomes a handle that provides continuity for successive calls to the PAM library. pam_start() expects arguments as follows: the service_name of the program, the username of the individual to be authenticated, a pointer to an application-supplied pam_conv structure, and a pointer to a pam_handle_t pointer.

An application must provide the conversation function used for direct communication between a loaded module and the application. The application also typically provides a means for the module to prompt the user for a password, etc.

The structure, pam_conv, is defined to be,

code
struct pam_conv {
    int (*conv) (int num_msg, const struct pam_message * *msg, struct pam_response * *resp, void *appdata_ptr);
    void *appdata_ptr;
};

© 2008 Linux Foundation
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>
<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>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>adler32</td>
<td>LSB</td>
</tr>
<tr>
<td>compress</td>
<td>LSB</td>
</tr>
<tr>
<td>compress2</td>
<td>LSB</td>
</tr>
<tr>
<td>compressBound</td>
<td>LSB</td>
</tr>
<tr>
<td>crc32</td>
<td>LSB</td>
</tr>
<tr>
<td>deflate</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateBound</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateCopy</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateEnd</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateInit2</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateInit</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateParams</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateReset</td>
<td>LSB</td>
</tr>
<tr>
<td>deflateSetDictionary</td>
<td>LSB</td>
</tr>
<tr>
<td>get_crc_table</td>
<td>LSB</td>
</tr>
<tr>
<td>gzclose</td>
<td>LSB</td>
</tr>
<tr>
<td>gzdopen</td>
<td>LSB</td>
</tr>
<tr>
<td>gzeof</td>
<td>LSB</td>
</tr>
<tr>
<td>gzerror</td>
<td>LSB</td>
</tr>
<tr>
<td>gzflush</td>
<td>LSB</td>
</tr>
<tr>
<td>gzgetc</td>
<td>LSB</td>
</tr>
<tr>
<td>gzgets</td>
<td>LSB</td>
</tr>
<tr>
<td>gzopen</td>
<td>LSB</td>
</tr>
<tr>
<td>gzprintf</td>
<td>LSB</td>
</tr>
<tr>
<td>gzputc</td>
<td>LSB</td>
</tr>
<tr>
<td>gzputs</td>
<td>LSB</td>
</tr>
<tr>
<td>gzread</td>
<td>LSB</td>
</tr>
<tr>
<td>gzwind</td>
<td>LSB</td>
</tr>
<tr>
<td>gzseek</td>
<td>LSB</td>
</tr>
<tr>
<td>gzsetparams</td>
<td>LSB</td>
</tr>
<tr>
<td>gztell</td>
<td>LSB</td>
</tr>
<tr>
<td>gzwrite</td>
<td>LSB</td>
</tr>
</tbody>
</table>
14 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

```c
#define Z_NULL 0
#define ZLIB_VERSION "1.2.2"
#define MAX_WBITS 15 /* 32K LZ77 window */
#define MAX_MEM_LEVEL 9 /* Maximum value for memLevel in deflateInit2 */
#define deflateInit2(strm,level,method,windowBits,memLevel,strategy)   
deflateInit2_((strm),(level),(method),(windowBits),(memLevel),(strategy),ZLIB_VERSION,sizeof(z_stream))
#define deflateInit2((strm),level)                                    
deflateInit2_((strm), (level), ZLIB_VERSION, sizeof(z_stream))
#define inflateInit2(strm,windowBits)                                 
inflateInit2_((strm), (windowBits), ZLIB_VERSION, sizeof(z_stream))
#define inflateInit(strm)                                           
inflateInit_((strm) ZLIB_VERSION, sizeof(z_stream))

typedef 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; /* next input byte */
    uInt avail_in; /* number of bytes available at next_in */
    uLong total_in; /* total nb of input bytes read so far */
    Bytef *next_out; /* next output byte should be put there */
    uInt avail_out; /* remaining free space at next_out */
    uLong total_out; /* total nb of bytes output so far */
    char *msg; /* last error message, NULL if no error */
    struct internal_state *state; /* not visible by applications */
    alloc_func zalloc; /* used to allocate the internal state */
    free_func zfree; /* used to free the internal state */
    voidpf opaque; /* private data object passed to zalloc and zfree */
    int data_type; /* best guess about the data type: ascii or binary */
    uLong adler; /* adler32 value of the uncompressed data */
    uLong reserved; /* reserved for future use */
} 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 uLong adler32(uLong adler, const Bytef * buf, uInt len);
extern int compress(Bytef * dest, uLongf * destLen, const Bytef * source,
    uLong sourceLen);
extern int compress2(Bytef * dest, uLongf * destLen, const Bytef * source,
    uLong sourceLen, int level);
extern uLong compressBound(uLong sourceLen);
extern uLong crc32(uLong crc, const Bytef * buf, uInt len);
extern int deflate(z_streamp strm, int flush);
extern int deflateCopy(z_streamp dest, z_streamp source);
extern int deflateEnd(z_streamp strm);
extern int deflateInit2_(z_streamp strm, int level, int method,
    int windowBits, int memLevel, int strategy,
    const char *version, int stream_size);
extern int deflateInit_(z_streamp strm, int level, const char *
    version,
    int stream_size);
extern int deflateParams(z_streamp strm, int level, int strategy);
extern int deflateReset(z_streamp strm);
extern int deflateSetDictionary(z_streamp strm, const Bytef *
    dictionary,
    uInt dictLength);
extern const uLongf *get_crc_table(void);
extern int gzclose(gzFile file);
extern gzFile gzdopen(int fd, const char *mode);
extern int gzeof(gzFile file);
extern const char *gzerror(gzFile file, int *errnum);
extern int gzflush(gzFile file, int flush);
extern char *gzgets(gzFile file, char *buf, int len);
extern int gzputc(gzFile file, int c);
extern int gzputs(gzFile file, const char *s);
extern int gzread(gzFile file, voidp buf, unsigned int len);
extern int gzrewind(gzFile file);
extern z_off_t gzseek(gzFile file, z_off_t offset, int whence);
extern int gzsetparams(gzFile file, int level, int strategy);
extern z_off_t gztell(gzFile file);
extern int gzwrite(gzFile file, voidp buf, unsigned int len);
extern int inflate(z_streamp strm, int flush);
extern int inflateCopy(z_streamp strm, z_streamp source);
extern int inflateEnd(z_streamp strm);
extern int inflateInit2_(z_streamp strm, int windowBits,
    const char *version, int stream_size);
extern int inflateInit_(z_streamp strm, int level, int method,
    const Bytef * dictionary,
    uInt dictLength);
extern int inflateSync(z_streamp strm);
extern int inflateSyncPoint(z_streamp z);
extern int uncompress(Bytef * dest, uLongf * destLen, const Bytef *
    source,
    uLong sourceLen);
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 Bytef * 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:

```c
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
#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

#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:

    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_streamp 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, 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**

```c
#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**
#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 ref-
ence 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 defla-
teEnd(). 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**

```c
#include <zlib.h>
int deflateEnd(z_streamp 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
#include <zlib.h>
int deflateInit2_ (z_stream 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.

\textbf{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.

\textbf{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_streamp 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_stream *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

#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.
deflateSetDictionary

Name

deflateSetDictionary — initialize compression dictionary

Synopsis

#include <zlib.h>
int deflateSetDictionary(z_stream * 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:
\[ x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^4 + x + 1 \]

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 gzdopen(), 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 gerror() 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.

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

#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 gdopen()).

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 accord-
ingly.

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 s to the compressed file referenced by 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 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 file to indicate the error. Applications should use gzerror() to access this error value. If file is NULL, gzputs() shall return Z_STREAM_ERR.

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.
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 gzopen()). 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**

```c
#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 gz-
dopen()). On entry, buf shall point to a buffer containing len bytes of uncom-
pressed 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 iden-
tified by file to indicate the error. Applications should use gzerror() to ac-
cess this error value.

Z_ERRNO

An underlying base library function has indicated an error. The global vari-
able 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.
**inflated**

**Name**

`inflate` — decompress data

**Synopsis**

```c
#include <zlib.h>
int inflate(z_streamp 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.
inflateEnd

Name

inflateEnd — free decompression stream state

Synopsis

#include <zlib.h>
int inflateEnd(z_stream *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.
inflatableInit2_

Name

inflatableInit2_ — initialize decompression system

Synopsis

#include <zlib.h>
int inflatableInit2_ (z_stream strm, int windowBits, char * version, int stream_size);

Description

The inflatableInit2_() 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, inflatableInit2_() 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 inflatableInit2_() function is not in the source standard; it is only in the binary standard. Source applications should use the inflatableInit2() macro.

Return Value

On success, the inflatableInit2_() function shall return Z_OK. Otherwise, inflatableInit2_() shall return a value as described below to indicate the error.

Errors

On error, inflatableInit2_() 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.
inflatableInit_

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

```
#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

#include <zlib.h>
int inflateSyncPoint(z_stream *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**

```c
#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**

```c
#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 libncurses

Table 14-3 defines the library name and shared object name for the libncurses 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 types of the following interfaces have had the const qualifier added as shown here, as compared to the specification in X/Open Curses.

```c
extern const char *keyname (int);
extern SCREEN *newterm (const char *, FILE *, FILE *);
extern const char *uncrl (chtype);

extern int mvprintw (int, int, const char *, ...);
extern int mwprintw (WINDOW *, int, int, const char *, ...);
extern int printw (const char *, ...);
extern int mvprintw (int, int, const char *, ...);
extern int mvwprintw (WINDOW *, int, int, const char *, ...);
extern int wprintw (WINDOW *, const char *, ...);
extern int mvscanw (int, int, const char *, ...);
extern int mwscanw (WINDOW *, int, int, const char *, ...);
extern int scanw (const char *, ...);
extern int vwprintw (WINDOW *, const char *, va_list);
extern int wprintw (WINDOW *, const char *, ...);

extern int mvwscanw (WINDOW *, int, int, const char *, ...);
extern int wscanw (WINDOW *, const char *, va_list);
```

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

[LSB] This Specification
[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>Function Name</td>
<td>Description</td>
<td>Library</td>
<td>Function Name</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td>chgat</td>
<td>SUS-CURSES</td>
<td>clear</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>clrtoeol</td>
<td>SUS-CURSES</td>
<td>color_content</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>curs_set</td>
<td>SUS-CURSES</td>
<td>def_prog_mode</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>delay_output</td>
<td>SUS-CURSES</td>
<td>delch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>delwin</td>
<td>SUS-CURSES</td>
<td>derwin</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>echo</td>
<td>SUS-CURSES</td>
<td>echochar</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>erasechar</td>
<td>SUS-CURSES</td>
<td>filter</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>getbkgd</td>
<td>SUS-CURSES</td>
<td>getch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>getwin</td>
<td>SUS-CURSES</td>
<td>halfdelay</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>has_il</td>
<td>SUS-CURSES</td>
<td>hline</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>immedok</td>
<td>SUS-CURSES</td>
<td>inch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>init_color</td>
<td>SUS-CURSES</td>
<td>init_pair</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>insch</td>
<td>SUS-CURSES</td>
<td>insdeltl</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>instr</td>
<td>SUS-CURSES</td>
<td>instr</td>
<td>LSB</td>
</tr>
<tr>
<td>is_wintouched</td>
<td>SUS-CURSES</td>
<td>isendwin</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>killchar</td>
<td>SUS-CURSES</td>
<td>leaveok</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>move</td>
<td>SUS-CURSES</td>
<td>mvaddch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>mvaddstr</td>
<td>SUS-CURSES</td>
<td>mvaddstr</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>mvdelch</td>
<td>SUS-CURSES</td>
<td>mvderwin</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>mvgetstr</td>
<td>SUS-CURSES</td>
<td>mvhline</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>mvinchstr</td>
<td>LSB</td>
<td>mvinnstr</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>mvinsnstr</td>
<td>SUS-CURSES</td>
<td>mvinnstr</td>
<td>LSB</td>
</tr>
<tr>
<td>mvline</td>
<td>SUS-CURSES</td>
<td>mvwaddch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>Function</td>
<td>Notes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwaddnstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwaddstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwchgat</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwdelch</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwgetch</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwgetnstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwgetstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwchgat</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwline</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwaddnstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwaddstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwchgat</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwdelch</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwgetch</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwgetnstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwgetstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwchgat</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mvwline</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>napms</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>newpad</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>newterm</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>newwin</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nl</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nocbreak</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nodelay</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noecho</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nonl</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noqiflush</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noraw</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>notimeout</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>overlay</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>overwrite</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pair_content</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pechochar</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pnoutrefresh</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prefresh</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>printw</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>putp</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>putwin</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>qiflush</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>raw</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>redrawwin</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>refresh</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset_prog_mode</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset_shell_mod</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>restarterm</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ripoffline</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>savetty</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scanw</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scr_dump</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scr_init</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scr_restore</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scr_set</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scroll</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scrollok</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_curterm</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setscreg</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setupterm</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_attroff</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_attron</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_attrset</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_clear</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_color</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_init</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_label</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_noutrefresh</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_refresh</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_restore</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_set</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>slk_touch</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>standend</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>standout</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>start_color</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subwin</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>syncok</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>termattr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>termname</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetent</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetflag</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetnum</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgoto</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetflag</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetnum</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tgetstr</td>
<td>[SUS-CURSES]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>timeout</td>
<td>SUS-CURSES</td>
<td>touchline</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>tputs</td>
<td>SUS-CURSES</td>
<td>typeahead</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>untouchwin</td>
<td>SUS-CURSES</td>
<td>use_env</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>vline</td>
<td>SUS-CURSES</td>
<td>vwprintw</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>vwscanw</td>
<td>LSB</td>
<td>waddch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>waddnstr</td>
<td>SUS-CURSES</td>
<td>wattr_set</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wattr_on</td>
<td>SUS-CURSES</td>
<td>wattr_set</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wattroff</td>
<td>SUS-CURSES</td>
<td>wattr_set</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wattrset</td>
<td>SUS-CURSES</td>
<td>wbkgd</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wchgt</td>
<td>SUS-CURSES</td>
<td>wclear</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wcorsyncup</td>
<td>SUS-CURSES</td>
<td>wdelch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wechochar</td>
<td>SUS-CURSES</td>
<td>wcurse</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wgetstr</td>
<td>SUS-CURSES</td>
<td>whline</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>winchstr</td>
<td>LSB</td>
<td>winnstr</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>winsnstr</td>
<td>SUS-CURSES</td>
<td>winsnstr</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wmove</td>
<td>SUS-CURSES</td>
<td>wnoutrefresh</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wrefresh</td>
<td>SUS-CURSES</td>
<td>wscanw</td>
<td>LSB</td>
</tr>
<tr>
<td>wstandend</td>
<td>SUS-CURSES</td>
<td>wstandout</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>wtimeout</td>
<td>SUS-CURSES</td>
<td>wtouchln</td>
<td>SUS-CURSES</td>
</tr>
</tbody>
</table>

An LSB conforming implementation shall provide the generic deprecated functions for Curses specified in Table 14-5, 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 14-5 libncurses - Curses Deprecated Function Interfaces**

<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>tgetent</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>tgetflag</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>tgetnum</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>tgetstr</td>
<td>SUS-CURSES</td>
</tr>
</tbody>
</table>
An LSB conforming implementation shall provide the generic data interfaces for Curses specified in Table 14-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-6 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['p'])
#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 __NCURSES_H     1
#define NCURSES_EXPORT(type)    type
#define NCURSES_EXPORT_VAR(type) type
#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_REVERSE A_REVERSE
#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;
typedef 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;                /* current cursor position */
    short _curx;
    short _maxy;                /* screen coords of upper-left-
window size */
    short _maxx;
    short _begy;                /* window state flags */
    short _begx;
    attr_t _attrs;               /* current attribute for non-
space character */
    ctype _bkgrnd;              /* current background
char/attribute pair */
    bool _notimeout;            /* no time out on function-key
entry? */
    bool _clear;                /* consider all data in the
window invalid? */
    bool _leaveok;              /* OK to not reset cursor on
exit? */
    bool _scroll;               /* OK to scroll this window? */
    bool _idlok;                /* OK to use insert/delete line?*/
    bool _idcok;                /* OK to use insert/delete char?*/
    bool _immed;                /* window in immed mode? (not yet
used) */
    bool _sync;                 /* window in sync mode? */
    bool _use_keypad;           /* process function keys into
KEY_ symbols? */
    int _delay;                 /* 0 = nodelay, <0 = blocking, >0
= delay */
    struct ldat *_line;        /* the actual line data */
    short _regtop;              /* top line of scrolling region*/
    short _regbottom;           /* bottom line of scrolling
region */
    int _parx;                  /* x coordinate of this window in
parent */
    int _pary;                  /* y coordinate of this window in
parent */
    WINDOW * _parent;           /* pointer to parent if a sub-
window */
    struct pdat _pad;           /* real begy is _begy + _yoffset*/
    cchar_t _bkgrnd;            /* current background
char/attribute pair */
};

#define KEY_F(n)        (KEY_F0+(n))
#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

© 2008 Linux Foundation
#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_RESET 0528
#define KEY_PRINT 0529
#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 0558
#define KEY_OPTIONS 0559
#define KEY_PREVIOUS 0560
#define KEY_REDO 0561
#define KEY_REFRESH 0562
#define KEY_REPLACE 0563
#define KEY_RESTART 0564
#define KEY_RESUME 0565
#define KEY_SAVE 0566
#define KEY_SBEG 0567
#define KEY_SCANCEL 0568
#define KEY_SCOMMAND 0569
#define KEY_SCOPY 0560
#define KEY_SCREATE 0561
#define KEY_SDC 0562
#define KEY_SDL 0600
#define KEY_SELECT 0601
#define KEY_SEND 0602
#define KEY_SEOL 0603
#define KEY_SEXIT 0604
#define KEY_SFIND 0605
#define KEY_SHELP 0606
#define KEY_SHOME 0607
#define KEY_SIC 0610
#define KEY_SLEFT 0611
ISO/IEC 23360 Part 1:2008(E)

#define KEY_SMESSAGE    0612
#define KEY_SMOVE       0613
#define KEY_SNEXT       0614
#define KEY_SOPTIONS    0615
#define KEY_SPREVIOUS   0616
#define KEY_SPRINT      0617
#define KEY_SREDO       0620
#define KEY_SREPLACE    0621
#define KEY_SRSUME      0622
#define KEY_SSAVE       0624
#define KEY_SSSAVE      0625
#define KEY_SUNDO       0626
#define KEY_SUSPEND     0627
#define KEY_UNDO        0630
#define KEY_MOUSE       0631
#define KEY_RESIZE      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_HORIZONTAL NCURSES_BITS(1UL,17)
#define A_LEFT NCURSES_BITS(1UL,18)
#define A_LOW NCURSES_BITS(1UL,19)
#define A_RIGHT NCURSES_BITS(1UL,20)
#define A_TOP NCURSES_BITS(1UL,21)
#define A_VERTICAL NCURSES_BITS(1UL,22)
#define A_STANDOUT NCURSES_BITS(1UL,8)
#define A_UNDERLINE NCURSES_BITS(1UL,9)
#define COLOR_PAIR(n)   NCURSES_BITS(n,0)
#define A_ATTRIBUTES NCURSES_BITS(~(1UL-1UL),0)

extern int COLORS;
extern int COLOR_PAIRS;
extern int COLS;
extern int LINES;
extern chtype acs_map[];
extern int addch(const chtype);
extern int addchnstr(const chtype *, int);
extern int addchstr(const chtype *);
extern int addnstr(const char *);
extern int addstr(const char *);
extern int attr_get(attr_t *, short *, void *);
extern int attr_off(attr_t, void *);
extern int attr_on(attr_t, void *);
extern int attr_set(attr_t, short, void *);
extern int attr_off(int);
extern int attron(int);
extern int attrset(int);
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, chtype,
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 mvaddnstr(int, int, const char *, int);
extern int mvaddstr(int, int, const char *);
extern int mvchgat(int, int, attr_t, short, const void *);
extern int mvcur(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 int mvhline(int, int, chtype, int);
extern chtype mvinch(int, int);
extern int mvinchnstr(int, int, chtype *, int);
extern int mvinchstr(int, 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 mvinsstr(int, int, char *);
extern int mvprintw(int, int, const char *, ...);
extern int mvscur(int, int, const char *, ...);
extern int mvline(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 mvwaddnstr(WINDOW *, int, int, const char *, int);
extern int mvwaddstr(WINDOW *, int, int, const char *);
extern int mvwchchgat(WINDOW *, int, int, attr_t, short, const void *);
extern int mvwdelch(WINDOW *, int, int);
extern int mvcgettch(WINDOW *, int, int);
extern int mvgetstr(WINDOW *, int, int, char *);
extern int mvhline(WINDOW *, int, int, chtype, int);
extern chtype mvwinch(WINDOW *, int, int);
extern int mvwchnstr(WINDOW *, int, int, chtype *, int);
extern int mvwchstr(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 mvwinsstr(WINDOW *, int, int, char *);
extern int mvwprintw(WINDOW *, int, int, const char *, ...);
extern int mvwscanw(WINDOW *, int, int, const char *, ...);
extern int mvwline(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(const char *, ...);
extern int putwin(WINDOW *, FILE *);
extern int raw(void);
extern int redrawwin(WINDOW *);
extern int refresh(void);
extern int reset_prog_mode(void);
extern int reset_shell_mode(void);
extern int ripoffline(int, int (*)(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 scr_restore(const char *);
extern int scr_set(const char *);
extern int scr1(int);
extern int scroll(WINDOW *);
extern int scrollok(WINDOW *, bool);
extern SCREEN *set_term(SCREEN *);
extern int slk_attr_set(const attr_t, short, void *);
extern int slk_attroff(const chtype);
extern int slk_attron(const chtype);
extern int slk_attrset(const chtype);
extern int slk_clear(void);
extern int slk_color(short);
extern int slk_init(int);
extern int 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 standend(void);
extern int standout(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 touchline(WINDOW *, int, int);
extern int touchwin(WINDOW *);
extern int typeahead(int);
extern const char *unctrl(chtype);
extern int ungetch(int);
extern int untouchwin(WINDOW *);
extern void use_env(bool);
extern int vidattr(chtype);
extern int vidputs(chtype, int (*)(int));
extern int vline(chtype, int);
extern int vwprintw(WINDOW *, const char *, va_list);
extern int vwscanf(WINDOW *, const char *, va_list);
extern int vwprintw(WINDOW *, const char *, va_list);
extern int vwscanf(WINDOW *, const char *, va_list);
extern int waddch(WINDOW *, const chtype);
extern int waddchnstr(WINDOW *, const chtype *, int);
extern int waddchstr(WINDOW *, const chtype *);
extern int waddnstr(WINDOW *, const char *, int);
14.6.2 term.h

extern TERMINAL *cur_term;
extern int del_curterm(TERMINAL *);
extern int putp(const char *);
extern int restarterm(char *, int, int *);
extern TERMINAL *set_curterm(TERMINAL *);
extern int setupterm(char *, int, int *);
extern int tgetent(char *, const char *);
extern int tgetflag(char *);
extern int tgetnum(char *);
14.7 Interface Definitions for libncurses

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

Other interfaces listed in Section 14.5 shall behave as described in the referenced base document.

inchnstr

Name

inchnstr — obtain a string of characters and their attributes from a curses window

Synopsis

```
#include <curses.h>
int inchnstr(chtype * chstr, int n);
```

Description

The interface inchnstr() shall behave as specified in X/Open Curses, except that inchnstr() shall return the number of characters that were read.

inchstr

Name

inchstr — obtain a string of characters and their attributes from a curses window

Synopsis

```
#include <curses.h>
int inchstr(chtype * chstr);
```

Description

The interface inchstr() shall behave as specified in X/Open Curses, except that inchstr() shall return the number of characters that were read.
instr

Name

instr — obtain a string of characters from a curses window

Synopsis

#include <curses.h>
int instr(char * str);

Description

The interface instr() shall behave as specified in X/Open Curses, except that instr() shall return the number of characters that were read.

mvcur

Name

mvcur — send cursor movement commands to terminal

Synopsis

#include <curses.h>
int mvcur(int oldrow, int oldcol, int newrow, int newcol);

Description

The interface mvcur() shall behave as described in X/Open Curses, except that if (newrow, newcol) is not a valid address for the terminal in use, the results of the mvcur() function are unspecified.

mvinchnstr

Name

mvinchnstr — obtain a string of characters and their attributes from a curses window

Synopsis

#include <curses.h>
int mvinchnstr(int y, int x, chtype * chstr, int n);

Description

The interface mvinchnstr() shall behave as specified in X/Open Curses, except that mvinchnstr() shall return the number of characters that were read.
**mvinchstr**

**Name**

*mvinchstr* — obtain a string of characters and their attributes from a curses window

**Synopsis**

```c
#include <curses.h>
int mvinchstr(int y, int x, chtype * chstr);
```

**Description**

The interface *mvinchstr()* shall behave as specified in *X/Open Curses*, except that *mvinchstr()* shall return the number of characters that were read.

**mvinstr**

**Name**

*mvinstr* — obtain a string of characters from a curses window

**Synopsis**

```c
#include <curses.h>
int mvinstr(int y, int x, char * str);
```

**Description**

The interface *mvinstr()* shall behave as specified in *X/Open Curses*, except that *mvinstr()* shall return the number of characters that were read.

**mvscanw**

**Name**

*mvscanw* — convert formatted input from a curses window

**Synopsis**

```c
#include <curses.h>
int mvscanw(int y, int x, const char *fmt, ...);
```

**Description**

The scanw family of functions shall behave as described in *X/Open Curses*, except as noted below.

**Differences**

This function returns *ERR* on failure. On success it returns the number of successfully matched and assigned input items. This differs from *X/Open Curses*, which indicates this function returns *OK* on success.
mvwinchnstr

Name

mvwinchnstr — obtain a string of characters and their attributes from a curses window

Synopsis

#include <curses.h>
int mvwinchnstr(WINDOW * win, int y, int x, chtype * chstr, int n);

Description

The interface mvwinchnstr() shall behave as specified in X/Open Curses, except that mvwinchnstr() shall return the number of characters that were read.

mvwinchstr

Name

mvwinchstr — obtain a string of characters and their attributes from a curses window

Synopsis

#include <curses.h>
int mvwinchstr(WINDOW * win, int y, int x, chtype * chstr);

Description

The interface mvwinchstr() shall behave as specified in X/Open Curses, except that mvwinchstr() shall return the number of characters that were read.

mvwinstr

Name

mvwinstr — obtain a string of characters from a curses window

Synopsis

#include <curses.h>
int mvwinstr(WINDOW * win, int y, int x, char * str);

Description

The interface mvwinstr() shall behave as specified in X/Open Curses, except that mvwinstr() shall return the number of characters that were read.
mvwscanw

Name

mvwscanw — convert formatted input from a curses window

Synopsis

#include <curses.h>
int mvwscanw(WINDOW *win, int y, int x, const char *fmt, ...);

Description

The scanw family of functions shall behave as described in X/Open Curses, except as noted below.

Differences

This function returns ERR on failure. On success it returns the number of successfully matched and assigned input items. This differs from X/Open Curses, which indicates this function returns OK on success.

ripoffline

Name

ripoffline — obtain a string of characters and their attributes from a curses window

Synopsis

#include <curses.h>
int ripoffline(int line, int (*init)(WINDOW *, int));

Description

The interface ripoffline() shall behave as specified in X/Open Curses, except that ripoffline() shall return -1 if the number of lines that were ripped off exceeds five.
scanw

Name

scanw — convert formatted input from a curses window

Synopsis

#include <curses.h>
int scanw(const char *fmt, ...);

Description

The scanw family of functions shall behave as described in X/Open Curses, except as noted below.

Differences

This function returns ERR on failure. On success it returns the number of successfully matched and assigned input items. This differs from X/Open Curses, which indicates this function returns OK on success.

vw_scanw

Name

vw_scanw — convert formatted input from a curses window

Synopsis

#include <curses.h>
int vw_scanw(WINDOW *win, const char *fmt, va_list vararglist);

Description

The scanw family of functions shall behave as described in X/Open Curses, except as noted below.

Differences

This function returns ERR on failure. On success it returns the number of successfully matched and assigned input items. This differs from X/Open Curses, which indicates this function returns OK on success.
vwscanw

Name
vwscanw — convert formatted input from a curses window

Synopsis
#include <curses.h>
int vw_scanw(WINDOW *win, const char *fmt, va_list vararglist);

Description
The scanw family of functions shall behave as described in X/Open Curses, except as noted below.

Differences
This function returns ERR on failure. On success it returns the number of successfully matched and assigned input items. This differs from X/Open Curses, which indicates this function returns OK on success.

winchnstr

Name
winchnstr — obtain a string of characters and their attributes from a curses window

Synopsis
#include <curses.h>
int winchnstr(WINDOW *win, chtype *chstr, int n);

Description
The interface winchnstr() shall behave as specified in X/Open Curses, except that winchnstr() shall return the number of characters that were read.

winchstr

Name
winchstr — obtain a string of characters and their attributes from a curses window

Synopsis
#include <curses.h>
int winchstr(WINDOW *win, chtype *chstr);

Description
The interface winchstr() shall behave as specified in X/Open Curses, except that winchstr() shall return the number of characters that were read.
winstr

Name

winstr — obtain a string of characters from a curses window

Synopsis

#include <curses.h>
int winstr(WINDOW *win, char *str);

Description

The interface winstr() shall behave as specified in ISO POSIX (2003), except that winstr() shall return the number of characters that were read.

wscanw

Name

wscanw — convert formatted input from a curses window

Synopsis

#include <curses.h>
int wscanw(WINDOW *win, const char *fmt, ...);

Description

The scanw family of functions shall behave as described in X/Open Curses, except as noted below.

Differences

This function returns ERR on failure. On success it returns the number of successfully matched and assigned input items. This differs from X/Open Curses, which indicates this function returns OK on success.

14.8 Interfaces for libutil

Table 14-7 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
14.8.1 Utility Functions

14.8.1.1 Interfaces for Utility Functions

An LSB conforming implementation shall provide the generic functions for Utility Functions specified in Table 14-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-8 libutil - Utility Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</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>

14.9 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 14.8 shall behave as described in the referenced base document.
forkpty

Name

forkpty — Create a new process attached to an available pseudo-terminal

Synopsis

#include <pty.h>
int forkpty(int *amaster, char *name, struct termios *termp, struct winsize *winp);

Description

The forkpty() function shall find and open a pseudo-terminal device pair in the same manner as the openpty() function. If a pseudo-terminal is available, forkpty() shall create a new process in the same manner as the fork() function, and prepares the new process for login in the same manner as login_tty().

If termp is not null, it shall refer to a termios structure that shall be used to initialize the characteristics of the slave device. If winp is not null, it shall refer to a winsize structure used to initialize the window size of the slave device.

Return Value

On success, the parent process shall return the process id of the child, and the child shall return 0. On error, no new process shall be created, -1 shall be returned, and errno shall be set appropriately. On success, the parent process shall receive the file descriptor of the master side of the pseudo-terminal in the location referenced by amaster, and, if name is not NULL, the filename of the slave device in name.

Errors

EAGAIN

Unable to create a new process.

ENOENT

There are no available pseudo-terminals.

ENOMEM

Insufficient memory was available.
login

Name
login — login utility function

Synopsis
#include <utmp.h>
void login (struct utmp * ut);

Description
The login() function shall update the user accounting databases. The ut parameter shall reference a utmp structure for all fields except the following:

1. The ut_type field shall be set to USER_PROCESS.
2. The ut_pid field shall be set to the process identifier for the current process.
3. The ut_line field shall be set to the name of the controlling terminal device. The name shall be found by 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**

```c
#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 *term, struct winsize *win);

Description

The openpty() function shall find an available pseudo-terminal and return file descriptors for the master and slave devices in the locations referenced by amaster and aslave respectively. If name is not NULL, the filename of the slave shall be placed in the user supplied buffer referenced by name. If term is not NULL, it shall point to a termios structure used to initialize the terminal parameters of the slave pseudo-terminal device. If win 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>
<thead>
<tr>
<th>Table 15-1 Commands And Utilities</th>
</tr>
</thead>
</table>
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>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>getopts</td>
<td>Get options</td>
<td>[1]</td>
</tr>
<tr>
<td>type</td>
<td>Determine type</td>
<td>[1]</td>
</tr>
<tr>
<td>umask</td>
<td>Set umask</td>
<td>[1]</td>
</tr>
<tr>
<td>command</td>
<td>Execute command</td>
<td>[1]</td>
</tr>
<tr>
<td>read</td>
<td>Read file</td>
<td>[1]</td>
</tr>
<tr>
<td>ulimit</td>
<td>Set resource limit</td>
<td>[1]</td>
</tr>
<tr>
<td>wait</td>
<td>Wait for process</td>
<td>[1]</td>
</tr>
</tbody>
</table>

---

**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.
**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)](http://example.com) but with differences as listed below.

**Differences**

**Options**

- `-d`
  
  is functionally equivalent to the `-r` option specified in [ISO POSIX (2003)](http://example.com).

- `-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)](http://example.com) but with differences as listed below.

**Differences**

Certain aspects of internationalized regular expressions are optional; see [Regular Expressions](http://example.com).
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**

```bash
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:
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.
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
dfile — determine file type

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

Differences
The -m, -h, -d, and -i options need not be supported.

fuser

Name
dfuser — identify processes using files or sockets

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

Differences
The dfuser command is a system administration utility, see Path For System Administration Utilities.

Option Differences

c
  has unspecified behavior.

-f
  has unspecified behavior.
gettext

Name

ggettext — retrieve text string from message catalog

Synopsis

ggettext [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 [-cdhflLnNrtV19] [-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 -1. 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 -n option is also specified, the uncompressed name, date and time are those stored within the compressed file, if present.

If the -q option is also specified, the title and totals lines are not displayed.

-L, --license

displays the gzip license and quit.

-n, --no-name

does not save the original file name and time stamp by default when compressing. (The original name is always saved if the name had to be truncated.) When decompressing, do not restore the original file name if present (remove only the gzip suffix from the compressed file name) and do not restore the original time stamp if present (copy it from the compressed file). This option is the default when decompressing.

-N, --name

always saves the original file name and time stamp when compressing; this is the default. When decompressing, restore the original file name and time stamp if present. This option is useful on systems which have a limit on file name length or when the time stamp has been lost after a file transfer.

-q, --quiet

suppresses all warnings.

-r, --recursive

travels the directory structure recursively. If any of the file names specified on the command line are directories, gzip will descend into the directory and compress all the files it finds there (or decompress them in the case of gunzip).

-S .suf, --sufix .suf

uses suffix .suf instead of .gz.

-t, --test

checks the compressed file integrity.

-v, --verbose

displays the name and percentage reduction for each file compressed or decompressed.

-#, --fast, --best

regulates the speed of compression using the specified digit #, where -1 or --fast indicates the fastest compression method (less compression) and -9 or --best indicates the slowest compression method (best compression). The default compression level is -6 (that is, biased towards high compression at expense of speed).
LSB Deprecated Options

The behaviors specified in this section are expected to disappear from a future version of the LSB; applications should only use the non-LSB-deprecated behaviors.

\(-V, \text{--version}\)

\text{displays the version number and compilation options, then quits.}

\textbf{hostname}

\textbf{Name}

\texttt{hostname} — show or set the system's host name

\textbf{Synopsis}

\texttt{hostname [name]}

\textbf{Description}

\texttt{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 \texttt{gethostname()} function.

When called with a \texttt{name} argument, and the user has appropriate privilege, the command sets the host name.

\textbf{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...
```

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 — off line print

Synopsis

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 device 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

© 2008 Linux Foundation
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 mkfifo command is a simpler interface for that purpose.
**mktemp**

**Name**

mktemp — make temporary file name (unique)

**Synopsis**

```bash
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 [-fnrsvw] [-t vfstype] [-o options] device dir
```

**Description**

As described in [ISO POSIX (2003)](https://en.wikipedia.org/wiki/ISO_POSIX), 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.

no user

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 new-line character. Empty lines, or lines containing only white-space, shall be ignored. Comments can in certain circumstances alter the behavior of msgfmt. See Comment Handling below for details on comment processing. The format of a statement is:

directive value

Each directive starts at the beginning of the line and is separated from value by white space (such as one or more space or tab characters). The value consists of one or more quoted strings separated by white space. If two or more strings are specified as value, they are normalized into single string using the string normalization syntax specified in ISO C (1999). The following directives are supported:

domain domainname
msgid message_identifier
msgid_plural untranslated_string_plural
msgstr message_string
msgstr[n] message_string

The behavior of the domain directive is affected by the options used. See OPTIONS for the behavior when the -o option is specified. If the -o option is not specified, the behavior of the domain directive is as follows:

1. All msgids from the beginning of each .po file to the first domain directive are put into a default message object file, messages (or messages.mo if the --strict option is specified).

2. When msgfmt encounters a domain domainname directive in the .po file, all following msgids until the next domain directive are put into the message object file domainname (or domainname.mo if --strict option is specified).

3. Duplicate msgids are defined in the scope of each domain. That is, a msgid is considered a duplicate only if the identical msgid exists in the same domain.

4. All duplicate msgids are ignored.

The msgid directive specifies the value of a message identifier associated with the directive that follows it. The msgid_plural directive specifies the plural form message specified to the plural message handling functions ngettext(), dngettext() or dcngettext(). The message_identifier string identifies a target string to be used at retrieval time. Each statement containing a msgid directive shall be followed by a statement containing a msgstr directive or msgstr[n] directives.

The msgstr directive specifies the target string associated with the message_identifier string declared in the immediately preceding msgid directive.

The msgstr[n] (where \( n = 0, 1, 2, \ldots \)) directive specifies the target string to be used with plural form handling functions ngettext(), dngettext() and dcngettext().

© 2008 Linux Foundation
Message strings can contain the following escape sequences:

Table 15-1 Escape Sequences
| \n | newline                  |
| \t | tab                     |
| \v | vertical tab            |
| \b | backspace               |
| \r | carriage return         |
| \f | formfeed                |
| \\ | backslash               |
| "  | double quote            |
| \ddd | octal bit pattern       |
| \xHH | hexadecimal bit pattern |
Comment Handling

Comments are introduced by a #, and continue to the end of the line. The second character (i.e. the character following the #) has special meaning. Regular comments should follow a space character. Other comment types include:

# normal-comments
#. automatic-comments
#: reference...
#, flag

Automatic and reference comments are typically generated by external utilities, and are not specified by the LSB. The \texttt{msgfmt} command shall ignore such comments.

\textbf{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 msgstr string might not be a correct translation. Only the translator (i.e. the individual undertaking the translation) can judge if the translation requires further modification, or is acceptable as is. Once satisfied with the translation, the translator then removes this fuzzy flag.

If this flag is specified, the \texttt{msgfmt} utility will not generate the entry for the immediately following msgid in the output message catalog, unless the \texttt{--use-fuzzy} is specified.

c-format
no-c-format

The c-format flag indicates that themsgid string is used as format string by printf()-like functions. If the c-format flag is given for a string the \texttt{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:

\texttt{nplurals=2; plural=n \textasciitilde 1 ? 0 : 1}

indicates that there are 2 plural forms in the language; msgstr[0] is used if \texttt{n \textasciitilde 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 code_set is used to indicate the codeset to be used to encode the message strings. If the output string’s codeset is different from the message string's codeset, codeset conversion from the message strings's codeset to the output string's codeset will be performed upon the call of gettext(), dgettext(), dcgettext(), ngettext(), dngettext(), and dcngettext(). The output string's codeset is determined by the current locale's codeset (the return value of nl_langinfo(CODESET)) by default, and can be changed by the call of bind_textdomain_codeset().

Exit Status
The following exit values are returned:

0
   Successful completion.

>0
   An error occurred.

Application Usage
Neither msgfmt nor any gettext() function imposes a limit on the total length of a message. Installing message catalogs under the C locale is pointless, since they are ignored for the sake of efficiency.

Examples
Example 1: Examples of creating message objects from message files.
In this example module1.po, module2.po and module3.po are portable message object files.
examp% cat module1.po
# default domain "messages"
msgid "message one"
msgstr "mensaje número uno"
#
domain "help_domain"
msgid "help two"
msgstr "ayuda número dos"
#
domain "error_domain"
msgid "error three"
msgstr "error número tres"
example% cat module2.po

# default domain "messages"
msgid "message four"
msgstr "mensaje número cuatro"
#
domain "error_domain"
msgid "error five"
msgstr "error número cinco"
#
domain "window_domain"
msgid "window six"
msgstr "ventana número seises"

example% cat module3.po

# default domain "messages"
msgid "message seven"
msgstr "mensaje número siete"

The following command will produce the output files messages, help_domain, and error_domain.

example% msgfmt module1.po

The following command will produce the output files messages.mo, help_domain.mo, error_domain.mo, and window_domain.mo.

example% msgfmt module1.po module2.po

The following example will produce the output file hello.mo.

example% msgfmt -o hello.mo module3.po
newgrp

**Name**
newgrp — change group ID

**Synopsis**
newgrp [group]

**Description**
The `newgrp` command is as specified in ISO POSIX (2003), but with differences as listed below.

**Differences**
The `-1` 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]

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.

-1

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)](https://www.unix.org). 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)](https://www.unix.org). 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.
seq

Name

seq — generate a sequence of numbers

Synopsis

/usr/bin/seq [-f fmt_str] [-s sep_str] [first_num] [inc_num] last_num

Description

The seq command shall output a sequence of numbers from first_num to last_num, stepping by the increment inc_num. The first_num and last_num parameters may be omitted, and default to 1 even when first_num is greater than last_num. Floating-point values may be specified for first_num, inc_num, and last_num.

The fmt_str parameter is a floating point format string like the one used for the printf() function in C.

The sep_str parameter string separates the values that are output. The default is a newline character (/n).

Note: If first_num is less than last_num and inc_num is negative, or first_num is greater than last_num and inc_num is positive, seq shall not generate any output.

Standard Options

-f fmt_str
Format the numbers in the output sequence according to fmt_str, a floating point format string like the one used for the printf() function in C.

-s sep_str
Separate the numbers in the output sequence with sep_str. The default separator string is a newline character (/n).

first_num
The first number in the output sequence. Defaults to 1. May be a floating point value.

inc_num
The increment for the output sequence. Defaults to 1. May be a floating point value.

last_num
The last number in the output sequence. May be a floating point value.
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()), 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] -a [-nr] [-t vfstype] 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 be-
haves 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 con-
cerning 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 sym-
boic 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.

© 2008 Linux Foundation
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 this specification relies, and where this specification requires a specific behavior.

Note: This specification 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 this specification does not require this behavior.

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_SETLK` 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 `1ink()` function shall not work across file systems, and shall fail and set `errno` to `EXDEV` as described as optional behavior in ISO POSIX (2003).

### 18.2 Optional Mandatory Behaviors

This section specifies behaviors that are mandatory in one of the standards on which this specification relies, but which are optional in this specification.

ISO POSIX (2003) describes the behavior of the file access time, available as the `st_atime` field of the `stat` and `stat64` structures. An LSB conforming implementation need not update this information every time a file is accessed.

**Note:** A subsequent edition of the POSIX standard no longer mandates updating of `st_atime` but the older edition is still the guiding standard for this specification, thus this exception is needed.

### 18.3 Executable Scripts

An executable script is an executable file of which the first two characters are `#!` as defined in the portable character set. In ISO POSIX (2003), this construct is undefined, but reserved for implementations which wish to provide this functionality. LSB conforming implementations shall support executable scripts.

A successful call to a function of the exec family with an executable script as the first parameter shall result in a new process, where the process image started is that of the interpreter. The path name of the interpreter follows the `#!` characters.

If the executable script has a first line

```
#!/interpreter [arg]
```

then `interpreter` shall be called with an argument array consisting of an unspecified zeroth argument, followed by `arg` (if present), followed by a path
name for the script, followed by the arguments following the zeroth argument in the exec call of the script.

The interpreter shall not perform any operations on the first line of an executable script.

The first line of the executable script shall meet all of the following criteria otherwise the results are unspecified:

1. Is one of the forms:

   ```
   #!interpreter
   #! interpreter
   #!interpreter_arg
   #! interpreter_arg
   ```

2. The `interpreter` argument is an absolute pathname of an executable file other than an executable script.

3. Neither the `interpreter` argument nor the `arg` argument, if present, contain any quoting characters.

4. Neither the `interpreter` argument nor the `arg` argument, if present, contain any whitespace characters.

5. The length of the entire line is no longer than 80 bytes.

If the interpreter is required by this specification to be in a specific named directory, a conforming application must use that path for `interpreter`, as implementations are not prohibited from having other, possibly non-conforming, versions of the same interpreter installed on the system. If the interpreter is a required command in this specification, but does not have a required path, the application should take special measures to insure the appropriate version is selected. If the interpreter is not a required command in this specification, the application must make appropriate provisions that the interpreter is available at the appropriate path.

**Note:** In case the path is not specified, it is recommended that an installation script for executable scripts use the standard `PATH` returned by a call to the `getconf` command with the argument `PATH`, combined with the `command` command to determine the location of a standard command.

For example to determine the location of the standard `awk` command:

```bash
PATH=`getconf PATH` command -v awk
```

The installation script should ensure that the returned pathname is an absolute pathname prior to use, since a shell builtin might be returned for some utilities.

Use of the common form `#!/usr/bin/env interpreter` is not recommended as the `PATH` will be unknown at execution time and an alternative version of `interpreter` might be selected.
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` command, 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:

- Range expression (such as `[a-z]`) can be based on code point order instead of collating element order.
- Equivalence class expression (such as `[=a=]`) and multi-character collating element expression (such as `[.ch.]`) are optional.
- Handling of a multi-character collating element is optional.

This affects at least the following utilities:

- `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

- `regexes` (see `regexes`)

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:
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 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 implementa-
tions 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 sym-
boлич 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/in-
20 System Initialization

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 func-
tionality. 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</td>
</tr>
<tr>
<td>3</td>
<td>normal/full multiuser</td>
</tr>
<tr>
<td>4</td>
<td>reserved for local use, default is normal/full</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.

stime

the system time has been set, for example by using a network-based time program such as `ntpq` or `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.

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).

20.8 Init Script Functions

Each conforming init script shall execute the commands in the file `/lib/lsb/init-functions` in the current environment (see shell special built-in command `dot`). This file shall cause the following shell script commands to be defined in an unspecified manner.

**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.

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. `set -e`) when sourcing this file, or calling any of the commands thus made available.

The `start_daemon`, `killproc` and `pidofproc` functions shall use the following algorithm for determining the status and the process identifiers of the specified program.

1. If the `-p pidfile` option is specified, and the named `pidfile` exists, a single line at the start of the `pidfile` shall be read. If this line contains one or more numeric values, separated by spaces, these values shall be used. If the `-p pidfile` option is specified and the named `pidfile` does not exist, the functions shall assume that the daemon is not running.

2. Otherwise, `/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.

**Note:** Commonly used methods check either for the existence of the `/proc/pid directory or use `/proc/pid/exe` and `/proc/pid/cmdline`. Relying only on `/proc/pid/exe` is discouraged since this specification does not specify the existence of, or semantics for, `/proc`. Additionally, using `/proc/pid/exe` may result in a not-running status for daemons that are written in a script language.
Conforming implementations may use other mechanisms besides those based on pidfiles, unless the 
-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 SIGHUP followed by a SIGKILL 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.

### log_warning_msg message

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>
<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 ID</td>
</tr>
<tr>
<td>daemon</td>
<td>daemon</td>
<td>Legacy User ID/Group ID</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>
<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>
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 22-1 RPM File Format

<table>
<thead>
<tr>
<th>Section</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead</td>
<td>unsigned char magic[4];</td>
</tr>
<tr>
<td>Signature</td>
<td>unsigned char major, minor;</td>
</tr>
<tr>
<td>Header</td>
<td>short type;</td>
</tr>
<tr>
<td>Payload</td>
<td>short archnum;</td>
</tr>
<tr>
<td></td>
<td>char name[66];</td>
</tr>
<tr>
<td></td>
<td>short osnum;</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

struct rpmlead {
    unsigned char magic[4];
    unsigned char major, minor;
    short type;
    short archnum;
    char name[66];
    short osnum;
}

© 2008 Linux Foundation
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>
<thead>
<tr>
<th>Table 22-2 Signature Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Record</td>
</tr>
<tr>
<td>Array of Index Records</td>
</tr>
<tr>
<td>Store of Index Values</td>
</tr>
</tbody>
</table>
22.2.2.1 Header Record

struct rpmheader {
    unsigned char magic[4];
    unsigned char reserved[4];
    int nindex;
    int hsize;
};

magic
Value identifying this record as an RPM header record. This value shall be "\216\255\350\001".

reserved
Reserved space. This value shall be "\000\000\000\000".

nindex
The number of Index Records that follow this Header Record. There should be at least 1 Index Record.

hsize
The size in bytes of the storage area for the data pointed to by the Index Records.

22.2.2.2 Index Record

struct rpmhdrindex {
    int tag;
    int type;
    int offset;
    int count;
};

tag
Value identifying the purpose of the data associated with this Index Record. 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_HDR18NTABLE index.

#### 22.2.2.2.2 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_HDR18NTABLE</td>
<td>100</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Optional</td>
</tr>
</tbody>
</table>

RPMTAG_HEADERSIGNATURES

The signature tag differentiates a signature header from a metadata header, and identifies the original contents of the signature header.
RPMTAG_HEADERIMMUTABLE

This tag contains an index record which specifies the portion of the Header Record which was used for the calculation of a signature. This data shall be preserved or any header-only signature will be invalidated.

RPMTAG_HEADERI18NTABLE

Contains a list of locales for which strings are provided in other parts of the package.

Not all Index records defined here will be present in all packages. Each tag value has a status which is defined here.

Required

This Index Record shall be present.

Optional

This Index Record may be present.

Informational

This Index Record may be present, but does not contribute to the processing of the package.

Deprecated

This Index Record should not be present.

Obsolete

This Index Record shall not be present.

Reserved

This Index Record shall not be present.

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>
<tr>
<td>RPMSIGTAG_PAYLOADSIZ E</td>
<td>1007</td>
<td>INT32</td>
<td>1</td>
<td>Optional</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_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>
</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_DSA</td>
<td>267</td>
<td>BIN</td>
<td>65</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMSIGTAG_RSA</td>
<td>268</td>
<td>BIN</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMSIGTAG_PGP</td>
<td>1002</td>
<td>BIN</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMSIGTAG_GPG</td>
<td>1005</td>
<td>BIN</td>
<td>65</td>
<td>Optional</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 tag 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_VERSION</td>
<td>1001</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_RELEASE</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_DESCRIPTION</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_DISTRIBUTION</td>
<td>1010</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_VENDOR</td>
<td>1011</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_LICENSE</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_GROUPE</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>
<tr>
<td>RPMTAG_ARCH</td>
<td>1022</td>
<td>STRING</td>
<td>1</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_SOURCERPM</td>
<td>1044</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_HIVESIZE</td>
<td>1046</td>
<td>INT32</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_RPM</td>
<td>1064</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
</tbody>
</table>
### 22 Software Installation

#### ISO/IEC 23360 Part 1:2008(E)

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION</td>
<td>1</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_COOKIE</td>
<td>1094</td>
<td>STRING</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_DISTRIBUT</td>
<td>1123</td>
<td>STRING</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_LOADFORMAT</td>
<td>1124</td>
<td>STRING</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_LOADCOMPRESSION</td>
<td>1125</td>
<td>STRING</td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_LOADFLAGS</td>
<td>1126</td>
<td>STRING</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_PACKAGER**

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_SOURCE RPM

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_PRE IN</td>
<td>1023</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_POS TIN</td>
<td>1024</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Table 22.2.4.2.3-1: Tag Reference Table

<table>
<thead>
<tr>
<th>Tag Type</th>
<th>Tag Code</th>
<th>Type</th>
<th>Length</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_PREUN</td>
<td>1025</td>
<td>STRING</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_POSUN</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 ac-
cess 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_FILENAMES</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_FILERDEVS</td>
<td>1033</td>
<td>INT16</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILETIMES</td>
<td>1034</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_FILMD5S</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_FILEDVICES</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_BASENAMES</td>
<td>1117</td>
<td>STRING_ARRAY</td>
<td></td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_DIRNAMES</td>
<td>1118</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_FILENAMES
This tag specifies the size of each file in the archive.

RPMTAG_FILEMODES
This tag specifies the mode of each file in the archive.

RPMTAG_FILERDEVS
This tag specifies the device number from which the file was copied.
RPMTAG_FILEMTIMES
This tag specifies the modification time in seconds since the epoch of each file in the archive.

RPMTAG_FILEMD5S
This tag specifies the ASCII representation of the MD5 sum of the corresponding file contents. This value is empty if the corresponding archive entry is not a regular file.

RPMTAG_FILELINKTOS
The target for a symlink, otherwise NULL.

RPMTAG_FILEFLAGS
This tag specifies the bit(s) to classify and control how files are to be installed. 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 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 22-11 File Flags

<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:

RPMFILE_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.

RPMFILE_DOC

The file contains documentation.

RPMFILE_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>
<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_REQUIREFLAGS</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_PROVIDENAME</td>
<td>1112</td>
<td>INT32</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_PROVIDENAME</td>
<td>1113</td>
<td>STRING_AR</td>
<td></td>
<td>Required</td>
</tr>
<tr>
<td>RPMTAG_PROVIDENAME</td>
<td>1114</td>
<td>INT32</td>
<td>1</td>
<td>Optional</td>
</tr>
<tr>
<td>RPMTAG_PROVIDENAME</td>
<td>1115</td>
<td>STRING_AR</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 RPMTAG_PROVIDENAME, RPMTAG_OBSOLETENAME or RPMTAG_PREREQ 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>
</tbody>
</table>
22 Software Installation
ISO/IEC 23360 Part 1:2008(E)

rpmlib(CompressedFileNames) 3.0.4-1 Indicates that the filenames in the Payload are represented in the RPMTAG_DIRINDEXES, RPMTAG_DIRNAME and RPMTAG_BASENAMES indexes. Optional

/bin/sh Optional
Interpretor usually required for installation scripts.

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>
<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_POST</td>
<td>0x400</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_SCRIPT_PRE_Un</td>
<td>0x800</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_SCRIPT_POST_Un</td>
<td>0x1000</td>
<td></td>
</tr>
<tr>
<td>RPMSENSE_RPMLIB</td>
<td>0x10000</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>TagValue</th>
<th>Type</th>
<th>Count</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_BUILDTIME</td>
<td>1006</td>
<td>INT32</td>
<td>1</td>
<td>Informational</td>
</tr>
<tr>
<td>RPMTAG_BUIHOST</td>
<td>1007</td>
<td>STRING</td>
<td>1</td>
<td>Informational</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
### RPMTAG_FILEVERIFYFLAGS

This tag specifies the bit(s) to control how files are to be verified after install, specifying which checks should be performed.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_FILEVERIFYFLAGS</td>
<td>1045</td>
<td>INT32</td>
<td>Optional</td>
</tr>
</tbody>
</table>

### RPMTAG_CHANGELOGTIME

This tag specifies the Unix time in seconds since the epoch associated with each entry in the Changelog file.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_CHANGELOGTIME</td>
<td>1080</td>
<td>INT32</td>
<td>Optional</td>
</tr>
</tbody>
</table>

### RPMTAG_CHANGELOGNAME

This tag specifies the name of who made a change to this package.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_CHANGELOGNAME</td>
<td>1081</td>
<td>STRING_AR</td>
<td>Optional</td>
</tr>
</tbody>
</table>

### RPMTAG_CHANGELOGTEXT

This tag specifies the changes associated with a changelog entry.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_CHANGELOGTEXT</td>
<td>1082</td>
<td>STRING_AR</td>
<td>Optional</td>
</tr>
</tbody>
</table>

### RPMTAG_OPTFLAGS

This tag indicates additional flags which may have been passed to the compiler when building this package.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_OPTFLAGS</td>
<td>1122</td>
<td>STRING</td>
<td>Informational</td>
</tr>
</tbody>
</table>

### RPMTAG_RHNPLATFORM

This tag contains an opaque string whose contents are undefined.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_RHNPLATFORM</td>
<td>1131</td>
<td>STRING</td>
<td>Deprecated</td>
</tr>
</tbody>
</table>

### RPMTAG_PLATFORM

This tag contains an opaque string whose contents are undefined.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPMTAG_PLATFORM</td>
<td>1132</td>
<td>STRING</td>
<td>Informational</td>
</tr>
</tbody>
</table>

### 22.2.5 Payload Section

The Payload section contains a compressed cpio archive. The format of this sec-
tion 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>
<thead>
<tr>
<th>Table 22-16 CPIO File Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPIO Header</strong></td>
</tr>
<tr>
<td>Filename</td>
</tr>
<tr>
<td>Padding</td>
</tr>
<tr>
<td>File data</td>
</tr>
<tr>
<td>Padding</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.

**c_mtime**

Value identifying the modification time of the file when it was read. This field shall match the corresponding value in the RPMTAG_FILEMTIMES index in the Header section.

**c_filesize**

Value identifying the size of the file. This field shall match the corresponding value in the RPMTAG_FILESIZES index in the Header section.

**c_devmajor**

The major number of the device containing the file system from which the file was read. With the exception of processing files with c_nlink >1, this field is ignored when installing a package. This field shall match the corresponding value in the RPMTAG_FILEDEVICES index in the Header section.

**c_devminor**

The minor number of the device containing the file system from which the file was read. With the exception of processing files with c_nlink >1, this field is ignored when installing a package. This field shall match the corresponding value in the RPMTAG_FILEDEVICES index in the Header section.

**c_rdevmajor**

The major number of the raw device containing the file system from which the file was read. This field is ignored when installing a package. This field shall match the corresponding value in the RPMTAG_RDEVS index in the Header section.

**c_rdevminor**

The minor number of the raw device containing the file system from which the file was read. This field is ignored when installing a package. This field shall match the corresponding value in the RPMTAG_RDEVS index in the Header section.
22 Software Installation

ISO/IEC 23360 Part 1:2008(E)

22.1 c_namesize

Value identifying the length of the filename, which is located immediately following the CPIO Header structure.

22.2 c_checksum

Value containing the CRC checksum of the file data. This field is not used, and shall contain the value "00000000". This field is ignored when installing a package.

A record with the filename "TRAILER!!" indicates the last record in the archive.

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-ssh-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.
ISO/IEC 23360 Part 1:2008(E)

Annex A Alphabetical Listing of Interfaces
A.1 libc
The behavior of the interfaces in this library is specified by the following Stan­
dards.
Large File Support [LFS]
This Specification [LSB]
RFC 1831/1832 RPC & XDR [RPC & XDR]
SUSv2 [SUSv2]
ISO POSIX (2003) [SUSv3]
POSIX 1003.1 2008 [SUSv4]
SVID Issue 3 [SVID.3]
SVID Issue 4 [SVID.4]
Table A­1 libc Function Interfaces
_Exit[SUSv3]

getcwd[SUSv3]

sched_rr_get_interval[S
USv3]

_IO_feof[LSB]

getdate[SUSv3]

sched_setaffinity(GLIB
C_2.3.4)[LSB]

_IO_getc[LSB]

getdelim[SUSv4]

sched_setparam[SUSv3
]

_IO_putc[LSB]

getdomainname[LSB]

sched_setscheduler[LS
B]

_IO_puts[LSB]

getdtablesize[LSB]

sched_yield[SUSv3]

__assert_fail[LSB]

getegid[SUSv3]

seed48[SUSv3]

__chk_fail(GLIBC_2.3.4
)[LSB]

getenv[SUSv3]

seed48_r[LSB]

__confstr_chk(GLIBC_2
.4)[LSB]

geteuid[SUSv3]

seekdir[SUSv3]

__ctype_b_loc(GLIBC_
2.3)[LSB]

getgid[SUSv3]

select[SUSv3]

__ctype_get_mb_cur_m
ax[LSB]

getgrent[SUSv3]

semctl[SUSv3]

__ctype_tolower_loc(G
LIBC_2.3)[LSB]

getgrent_r[LSB]

semget[SUSv3]

__ctype_toupper_loc(G
LIBC_2.3)[LSB]

getgrgid[SUSv3]

semop[SUSv3]

__cxa_atexit[LSB]

getgrgid_r[SUSv3]

send[SUSv4]

__cxa_finalize[LSB]

getgrnam[SUSv3]

sendfile[LSB]

__errno_location[LSB]

getgrnam_r[SUSv3]

sendfile64(GLIBC_2.3)
[LSB]

__fgets_chk(GLIBC_2.4
)[LSB]

getgrouplist[LSB]

sendmsg[SUSv4]

__fgets_unlocked_chk(
GLIBC_2.4)[LSB]

getgroups[SUSv3]

sendto[SUSv4]

© 2008 Linux Foundation

631


<table>
<thead>
<tr>
<th>Function</th>
<th>Notes</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>__fgetws_chk(GLIBC_2.4) [LSB]</td>
<td>gethostbyaddr[SUSv3]</td>
<td>setbuf[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fgetws_unlocked_chk(GLIBC_2.4) [LSB]</td>
<td>gethostbyaddr_r[LSB]</td>
<td>setbuffer[LSB]</td>
<td></td>
</tr>
<tr>
<td>__fpending [LSB]</td>
<td>gethostbyname[SUSv3]</td>
<td>setcontext[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fprintf_chk [LSB]</td>
<td>gethostbyname2[LSB]</td>
<td>setegid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fwprintf_chk(GLIBC_2.4) [LSB]</td>
<td>gethostbyname2_r[LSB]</td>
<td>seteuid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fxstat [LSB]</td>
<td>gethostid[SUSv3]</td>
<td>setgid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fxstat64[LSB]</td>
<td>gethostid[SUSv3]</td>
<td>setgid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fxstatat(GLIBC_2.4) [LSB]</td>
<td>gethostname[SUSv3]</td>
<td>setgrent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__fxstatat64(GLIBC_2.4) [LSB]</td>
<td>getitimer[SUSv3]</td>
<td>setgroups[LSB]</td>
<td></td>
</tr>
<tr>
<td>__getcwd_chk(GLIBC_2.4) [LSB]</td>
<td>getline[SUSv4]</td>
<td>sethostname[LSB]</td>
<td></td>
</tr>
<tr>
<td>__getgroups_chk(GLIBC_C_2.4) [LSB]</td>
<td>getloadavg[LSB]</td>
<td>setitimer[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__gethostname_chk(GLIBC_2.4) [LSB]</td>
<td>getlogin[SUSv3]</td>
<td>setlocale[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__getlogin_r_chk(GLIBC_2.4) [LSB]</td>
<td>getlogin_r[SUSv3]</td>
<td>setlogmask[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__getpagesize [LSB]</td>
<td>getnameinfo[SUSv3]</td>
<td>setpgid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__getpgid [LSB]</td>
<td>getopt[LSB]</td>
<td>setpgrp[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__h_errno_location [LSB]</td>
<td>getopt_long[LSB]</td>
<td>setpriority[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isinf[LSB]</td>
<td>getopt_long_only[LSB]</td>
<td>setprotoent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isnff[LSB]</td>
<td>getpagesize[LSB]</td>
<td>setpwnent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__ isnfl [LSB]</td>
<td>getpeertname[SUSv3]</td>
<td>setregid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isnan[LSB]</td>
<td>getpid[SUSv3]</td>
<td>setreuid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isnfl [LSB]</td>
<td>getpid[SUSv3]</td>
<td>setrlimit[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__isnfl [LSB]</td>
<td>getpid[SUSv3]</td>
<td>setrlimit64[LFS]</td>
<td></td>
</tr>
<tr>
<td>__libc_current_sigrtmin[LSB]</td>
<td>getppid[SUSv3]</td>
<td>setservent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__libc_start_main[LSB]</td>
<td>getprotobycname[SUSv3]</td>
<td>setsockopt[LSB]</td>
<td></td>
</tr>
<tr>
<td>__lxstat [LSB]</td>
<td>getprotobynumber[SUSv3]</td>
<td>setstate[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__lxstat64[LSB]</td>
<td>getprotobynumber_r[SUSv3]</td>
<td>setstate_r[LSB]</td>
<td></td>
</tr>
<tr>
<td>__mbsnrtowcs_chk(GLIBC_2.4) [LSB]</td>
<td>getprotobynumber_r[LSB]</td>
<td>setuid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Libraries</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><code>__mbsrtowcs_chk(GLIBC_2.4)[LSB]</code></td>
<td>getprotoent[SUSv3]</td>
<td>setutent[LSB]</td>
<td></td>
</tr>
<tr>
<td><code>__mbsrtowcs_chk(GLIBC_2.4)[LSB]</code></td>
<td>getprotoent_r[LSB]</td>
<td>setutxent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__memcpy_chk(GLIBC_2.3.4)[LSB]</code></td>
<td>getpwent[SUSv3]</td>
<td>setvbuf[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__mempcpy[LSB]</code></td>
<td>getpwent_r[LSB]</td>
<td>shmat[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__mempcpy_chk(GLIBC_2.3.4)[LSB]</code></td>
<td>getpwnam[SUSv3]</td>
<td>shmdt[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__memset_chk(GLIBC_2.3.4)[LSB]</code></td>
<td>getpwuid[SUSv3]</td>
<td>shmget[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__pread64_chk(GLIBC_2.4)[LSB]</code></td>
<td>getpwuid_r[SUSv3]</td>
<td>shutdown[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__readchk(GLIBC_2.4)[LSB]</code></td>
<td>getrlimit[SUSv3]</td>
<td>sigaction[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__printf_chk[LSB]</code></td>
<td>getrlimit64[LFS]</td>
<td>sigaddset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__rawmemchr[LSB]</code></td>
<td>getrusage[SUSv3]</td>
<td>sigaltsign[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__read_chk(GLIBC_2.4)[LSB]</code></td>
<td>getservbyname[SUSv3]</td>
<td>sigandset[LSB]</td>
<td></td>
</tr>
<tr>
<td><code>__readlink_chk(GLIBC_2.4)[LSB]</code></td>
<td>getservbyname_r[LSB]</td>
<td>sigdelset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__realpath_chk(GLIBC_2.4)[LSB]</code></td>
<td>getservbyport[SUSv3]</td>
<td>sigemptyset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__recv_chk(GLIBC_2.4)[LSB]</code></td>
<td>getservbyport_r[LSB]</td>
<td>sigfillset[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__recvfrom_chk(GLIBC_2.4)[LSB]</code></td>
<td>getservent[SUSv3]</td>
<td>sighold[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__register_atfork(GLIBC_2.3.2)[LSB]</code></td>
<td>getservent_r[LSB]</td>
<td>sigignore[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__sigsetjmp[LSB]</code></td>
<td>getsid[SUSv3]</td>
<td>siginterrupt[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__snprintf_chk[LSB]</code></td>
<td>gettimezone[SUSv3]</td>
<td>sigemptyset[LSB]</td>
<td></td>
</tr>
<tr>
<td><code>__strcat_chk(GLIBC_2.3.4)[LSB]</code></td>
<td>getsockname[SUSv3]</td>
<td>sigprocmask[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__strcat_chk(GLIBC_2.4)[LSB]</code></td>
<td>getsockopt[LSB]</td>
<td>sigsetjmp[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__strlen_chk(GLIBC_2.4)[LSB]</code></td>
<td>getsubopt[SUSv3]</td>
<td>siglongjmp[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__strcpy_chk(GLIBC_2.3.4)[LSB]</code></td>
<td>gettext[LSB]</td>
<td>signal[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__strcpy_chk(GLIBC_2.4)[LSB]</code></td>
<td>gettimeofday[LSB]</td>
<td>sigevent[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__strcpy_chk(GLIBC_2.4)[LSB]</code></td>
<td>getuid[SUSv3]</td>
<td>sigpause[LSB]</td>
<td></td>
</tr>
<tr>
<td><code>__strcat_chk(GLIBC_2.3.4)[LSB]</code></td>
<td>getutent[LSB]</td>
<td>sigpending[SUSv3]</td>
<td></td>
</tr>
<tr>
<td><code>__strcat_chk(GLIBC_2.4)[LSB]</code></td>
<td>getutent_r[LSB]</td>
<td>sigprocmask[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>__strdup</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__strncat_chk(GLIBC_2 3.4)[LSB]</td>
<td>getutxid[SUSv3]</td>
<td>sigqueue[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__strncpy_chk(GLIBC_2.3.4)[LSB]</td>
<td>getutxline[SUSv3]</td>
<td>sigrelse[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__strtok</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__strtol_internal</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__strtof_internal</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__strtold_internal</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__strtoull_internal</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__strtoul_internal</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__swprintf_chk(GLIBC_2.4)</td>
<td>globfree[SUSv3]</td>
<td>socket[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__vfprintf_chk</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__vfwprintf_chk</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__vprintf_chk</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__vsnprintf_chk</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__vswprintf_chk</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcpcpy_chk(GLIBC_2.4)</td>
<td>hsearch[SUSv3]</td>
<td>statfs[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__wcrtomb_chk</td>
<td>[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcscpy_chk(GLIBC_2.4)</td>
<td>hsearch_r[LSB]</td>
<td>statfs64[LSB]</td>
<td></td>
</tr>
<tr>
<td>__wcpncpy_chk(GLIBC_2.4)</td>
<td>htonl[SUSv3]</td>
<td>statvfs[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__wcscpy_chk(GLIBC_2.4)</td>
<td>htons[SUSv3]</td>
<td>statvfs64[LFS]</td>
<td></td>
</tr>
<tr>
<td>__wcscpy_chk(GLIBC_2.4)</td>
<td>iconv[SUSv3]</td>
<td>stime[LSB]</td>
<td></td>
</tr>
<tr>
<td>__wcpcpy_chk(GLIBC_2.4)</td>
<td>iconv_close[SUSv3]</td>
<td>stpcpy[LSB]</td>
<td></td>
</tr>
<tr>
<td>__wcrtomb_chk(GLIBC_2.4)</td>
<td>iconv_open[SUSv3]</td>
<td>stpcpy[LSB]</td>
<td></td>
</tr>
<tr>
<td>__wscat_chk(GLIBC_2.4)</td>
<td>if_freenameindex[SUSv3]</td>
<td>strcasecmp[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>__wscpy_chk(GLIBC_2.4)</td>
<td>if_indextoname[SUSv3]</td>
<td>strcasestr[LSB]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcsncat_chk(GLIBC_2.4) [LSB]</td>
<td>if_nameindex[SUSv3] strcat[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcsncpy_chk(GLIBC_2.4) [LSB]</td>
<td>if_nametoindex[SUSv3] strchr[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcsnrtombs_chk(GLIBC_2.4) [LSB]</td>
<td>imaxabs[SUSv3] strcmp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcsrtombs_chk(GLIBC_2.4) [LSB]</td>
<td>imaxdiv[SUSv3] strcoll[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcstod_internal [LSB]</td>
<td>index[SUSv3] strcpy[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcstof_internal [LSB]</td>
<td>inet_addr[SUSv3] strcsn[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcstol_internal [LSB]</td>
<td>inet_aton[LSB] strdup[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcstoul_internal [LSB]</td>
<td>inet_ntoa[SUSv3] strerror[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcstombs_chk(GLIBC_2.4) [LSB]</td>
<td>inet_ntop[SUSv3] strerror_r[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wctomb_chk(GLIBC_2.4) [LSB]</td>
<td>initgroups[LSB] strftime[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wcstold_internal [LSB]</td>
<td>initstate[LSB] strlen[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wmemcpy_chk(GLIBC_2.4) [LSB]</td>
<td>initstate_r[LSB] strncasecmp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wmemmove_chk(GLIBC_2.4) [LSB]</td>
<td>inotify_add_watch(GLIBC_2.4) [LSB] strncat[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wmemcpyp_chk(GLIBC_2.4) [LSB]</td>
<td>inotify_init(GLIBC_2.4) [LSB] strncmp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wmknod [LSB]</td>
<td>inotify_rm_watch(GLIBC_2.4) [LSB] strncpy[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__wmknodat(GLIBC_2.4) [LSB]</td>
<td>isalnum[SUSv3] strpbrk[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xpg_basename[LSB]</td>
<td>isalpha[SUSv3] strptime[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xpg_sigpause[LSB]</td>
<td>isascii[SUSv3] strsep[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xpg_strerror_r(GLIBC_2.4) [LSB]</td>
<td>isasciic[SUSv3] strftime[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xstat64[LSB]</td>
<td>isatty[SUSv3] strsep[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_longjmp[SUSv3]</td>
<td>iscntrl[SUSv3] strspn[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>_toupper[SUSv3]</td>
<td>isprint[SUSv3] strtoimax[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a64l[SUSv3]</td>
<td>ispunct[SUSv3] strtok[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>-------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>abort</td>
<td>SUSEv3</td>
<td>isspace</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>abs</td>
<td>SUSEv3</td>
<td>isupper</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>accept</td>
<td>SUSEv3</td>
<td>iswalnum</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>access</td>
<td>SUSEv3</td>
<td>iswalpha</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>acct</td>
<td>LSB</td>
<td>iswblank</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>adjtime</td>
<td>LSB</td>
<td>iswcntrl</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>alarm</td>
<td>SUSEv3</td>
<td>iswctype</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>alphasort</td>
<td>SUSEv4</td>
<td>iswdigit</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>alphasort64</td>
<td>LSB</td>
<td>iswgraph</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>asctime</td>
<td>SUSEv3</td>
<td>iswlower</td>
<td>SUSEv3</td>
</tr>
<tr>
<td>asctime_r</td>
<td>SUSEv3</td>
<td>iswprint</td>
<td>svc_getreqset[SVID.3]</td>
</tr>
<tr>
<td>asprintf</td>
<td>LSB</td>
<td>iswpunct</td>
<td>svc_register[LSB]</td>
</tr>
<tr>
<td>atof</td>
<td>SUSEv3</td>
<td>iswspace</td>
<td>svc_run[LSB]</td>
</tr>
<tr>
<td>atoi</td>
<td>SUSEv3</td>
<td>iswupper</td>
<td>svc_sendreply[LSB]</td>
</tr>
<tr>
<td>atoll</td>
<td>SUSEv3</td>
<td>iswdxigit</td>
<td>svcerr_auth[SVID.3]</td>
</tr>
<tr>
<td>authnone_create</td>
<td>SVID.4</td>
<td>jrand48</td>
<td>svcerr_decode[SVID.3]</td>
</tr>
<tr>
<td>basename</td>
<td>LSB</td>
<td>jrand48_r</td>
<td>svcerr_noprog[SVID.3]</td>
</tr>
<tr>
<td>bcmp</td>
<td>SUSEv3</td>
<td>key_decryptsession[SVID.3]</td>
<td></td>
</tr>
<tr>
<td>bcopy</td>
<td>SUSEv3</td>
<td>kill</td>
<td>svcerr_systemerr[SVID.3]</td>
</tr>
<tr>
<td>bind</td>
<td>SUSEv3</td>
<td>killpg</td>
<td>svcerrWeakauth[SVID.3]</td>
</tr>
<tr>
<td>bind_textdomain_code set</td>
<td>LSB</td>
<td>l64a[SUSEv3]</td>
<td>svcfd_create[RPC &amp; XDR]</td>
</tr>
<tr>
<td>bindresvport</td>
<td>LSB</td>
<td>labs[SUSEv3]</td>
<td>svcraw_create[RPC &amp; XDR]</td>
</tr>
<tr>
<td>bindtextdomain</td>
<td>LSB</td>
<td>lchown</td>
<td>svctcp_create[LSB]</td>
</tr>
<tr>
<td>brk</td>
<td>SUSEv2</td>
<td>lcong48</td>
<td>svcuda_p_create[LSB]</td>
</tr>
<tr>
<td>bsd_signal</td>
<td>SUSEv3</td>
<td>lcong48_r</td>
<td>swab[SUSEv3]</td>
</tr>
<tr>
<td>bsearch</td>
<td>SUSEv3</td>
<td>ldiv[SUSEv3]</td>
<td>swapcontext[SUSEv3]</td>
</tr>
<tr>
<td>btowc</td>
<td>SUSEv3</td>
<td>lfind[SUSEv3]</td>
<td>swprintf[SUSEv3]</td>
</tr>
<tr>
<td>bzero</td>
<td>SUSEv3</td>
<td>link[LSB]</td>
<td>swscanf[LSB]</td>
</tr>
<tr>
<td>calloc</td>
<td>SUSEv3</td>
<td>linkat(GLIBC_2.4)</td>
<td>symlink[SUSEv3]</td>
</tr>
<tr>
<td>callrpc</td>
<td>RPC &amp; XDR</td>
<td>listen[SUSEv3]</td>
<td>symlinkat(GLIBC_2.4)[SUSEv4]</td>
</tr>
<tr>
<td>catclose</td>
<td>SUSEv3</td>
<td>llabs[SUSEv3]</td>
<td>sync[SUSEv3]</td>
</tr>
<tr>
<td>catgets</td>
<td>SUSEv3</td>
<td>lldiv[SUSEv3]</td>
<td>sysconf[LSB]</td>
</tr>
<tr>
<td>catopen</td>
<td>SUSEv3</td>
<td>localeconv[SUSEv3]</td>
<td>syslog[SUSEv3]</td>
</tr>
<tr>
<td>Function</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfgetispeed[SUSv3]</td>
<td>localtime[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfgetospeed[SUSv3]</td>
<td>localtime_r[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfmakeraw[LSB]</td>
<td>lockf[SUSv3]</td>
<td>tcdrain[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>cfsetspeed[SUSv3]</td>
<td>lockf64[LFS]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfsetospeed[SUSv3]</td>
<td>longjmp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfsetspeed[LSB]</td>
<td>lrang48[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chdir[SUSv3]</td>
<td>lrang48_r[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chmod[SUSv3]</td>
<td>lseek[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chown[SUSv3]</td>
<td>lsearch[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chroot[SUSv2]</td>
<td>mbrlen[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clearerr[SUSv3]</td>
<td>mbstrlen[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clearerr_unlocked[LSB]</td>
<td>mbstrlen[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_create[SVID.4]</td>
<td>mbstrlen[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_pcreateerror[SVID.4]</td>
<td>mbstrlen[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_perror[SVID.4]</td>
<td>mbsinit[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_perror[SVID.4]</td>
<td>mbsinit[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_spcreateerror[SVI D.4]</td>
<td>mbsnrtowcs[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_sperror[SVID.4]</td>
<td>mbsnrtowcs[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clnt_raw_create[RPC &amp; XDR]</td>
<td>memccpy[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clntcp_create[RPC &amp; XDR]</td>
<td>memchr[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clntudp_bufcreate[RPC &amp; XDR]</td>
<td>memccpy[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>connect[SUSv3]</td>
<td>memmove[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>confstr[SUSv3]</td>
<td>memcpy[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creat[SUSv3]</td>
<td>mkdtemp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creat64[LFS]</td>
<td>mkfifo[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctermid[SUSv3]</td>
<td>mkfifoat(GLIBC_2.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime[SUSv3]</td>
<td>mkstemp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime_r[SUSv3]</td>
<td>mkstemp64[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock[SUSv3]</td>
<td>memmem[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>close[SUSv3]</td>
<td>memmove[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>closedir[SUSv3]</td>
<td>memrchr[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>closelog[SUSv3]</td>
<td>memset[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>confstr[SUSv3]</td>
<td>mkdir[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>connect[SUSv3]</td>
<td>mkdtempat(GLIBC_2.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>confstr[SUSv3]</td>
<td>mkstemp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime[SUSv3]</td>
<td>mkstemp64[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctermid[SUSv3]</td>
<td>mkfifoat(GLIBC_2.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime_r[SUSv3]</td>
<td>mkfifoat(GLIBC_2.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>confstr[SUSv3]</td>
<td>mkstemp[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime_r[SUSv3]</td>
<td>mkstemp64[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
<td>--------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>userid</td>
<td>SUSv2</td>
<td>mktemp</td>
<td>SUSv3</td>
</tr>
<tr>
<td>daemon</td>
<td>LSB</td>
<td>mkttime</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dcgettext</td>
<td>LSB</td>
<td>mlock</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dcngettext</td>
<td>SUSv3</td>
<td>mlockall</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ggettext</td>
<td>LSB</td>
<td>mremap</td>
<td>SUSv3</td>
</tr>
<tr>
<td>diff</td>
<td>SUSv3</td>
<td>mremap64</td>
<td>LFS</td>
</tr>
<tr>
<td>dirfd</td>
<td>SUSv4</td>
<td>mprotect</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dirname</td>
<td>SUSv3</td>
<td>mrand48</td>
<td>SUSv3</td>
</tr>
<tr>
<td>div</td>
<td>SUSv3</td>
<td>mrand48_r</td>
<td>LSB</td>
</tr>
<tr>
<td>dngettext</td>
<td>LSB</td>
<td>mprotect</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dprintf</td>
<td>SUSv4</td>
<td>msgctl</td>
<td>SUSv3</td>
</tr>
<tr>
<td>drand48</td>
<td>SUSv3</td>
<td>msgget</td>
<td>SUSv3</td>
</tr>
<tr>
<td>drand48_r</td>
<td>LSB</td>
<td>msgrcv</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dup</td>
<td>SUSv3</td>
<td>msgrcv</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dup2</td>
<td>SUSv3</td>
<td>msync</td>
<td>SUSv3</td>
</tr>
<tr>
<td>duplocale</td>
<td>SUSv3</td>
<td>munlock</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ecvt</td>
<td>SUSv3</td>
<td>munlockall</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endgrent</td>
<td>SUSv3</td>
<td>munmap</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endprotoent</td>
<td>SUSv3</td>
<td>nanosleep</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endpwent</td>
<td>SUSv3</td>
<td>newlocale</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endservent</td>
<td>SUSv3</td>
<td>nftw</td>
<td>SUSv3</td>
</tr>
<tr>
<td>endutent</td>
<td>LSB</td>
<td>nftw64</td>
<td>SUSv3</td>
</tr>
<tr>
<td>enduxtent</td>
<td>SUSv3</td>
<td>ngetext</td>
<td>LSB</td>
</tr>
<tr>
<td>epoll_create</td>
<td>SUSv3</td>
<td>nice</td>
<td>SUSv3</td>
</tr>
<tr>
<td>epoll_ctl</td>
<td>SUSv3</td>
<td>nl_langinfo</td>
<td>SUSv3</td>
</tr>
<tr>
<td>epoll_wait</td>
<td>SUSv3</td>
<td>nrand48</td>
<td>SUSv3</td>
</tr>
<tr>
<td>erand48</td>
<td>SUSv3</td>
<td>nrand48_r</td>
<td>LSB</td>
</tr>
<tr>
<td>erand48_r</td>
<td>LSB</td>
<td>ntohl</td>
<td>SUSv3</td>
</tr>
<tr>
<td>err</td>
<td>LSB</td>
<td>ntohs</td>
<td>SUSv3</td>
</tr>
<tr>
<td>error</td>
<td>LSB</td>
<td>open</td>
<td>SUSv3</td>
</tr>
<tr>
<td>errx</td>
<td>LSB</td>
<td>open_memstream</td>
<td>SUSv4</td>
</tr>
<tr>
<td>execl</td>
<td>SUSv3</td>
<td>open_wmemstream</td>
<td>SUSv4</td>
</tr>
<tr>
<td>execle</td>
<td>SUSv3</td>
<td>openat</td>
<td>SUSv4</td>
</tr>
<tr>
<td>remote</td>
<td>SUSv3</td>
<td>remote</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>SUSv4</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>execlp</td>
<td>openat64(GLIBC_2.4)</td>
<td>wait4[LSB]</td>
<td></td>
</tr>
<tr>
<td>execv</td>
<td>opendir[SUSv3]</td>
<td>waitid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>execve</td>
<td>openlog[SUSv3]</td>
<td>waitpid[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>execvp</td>
<td>pathconf[SUSv3]</td>
<td>warn[LSB]</td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>pause[SUSv3]</td>
<td>warnx[LSB]</td>
<td></td>
</tr>
<tr>
<td>faccessat(GLIBC_2.4)</td>
<td>pclose[SUSv3]</td>
<td>wcpcpy[LSB]</td>
<td></td>
</tr>
<tr>
<td>fchdir</td>
<td>perror[SUSv3]</td>
<td>wcpcpy[LSB]</td>
<td></td>
</tr>
<tr>
<td>chmod</td>
<td>pipe[SUSv3]</td>
<td>wctomb[LSB]</td>
<td></td>
</tr>
<tr>
<td>chmodat(GLIBC_2.4)</td>
<td>pmap_getport[LSB]</td>
<td>wcasecmp[LSB]</td>
<td></td>
</tr>
<tr>
<td>chown</td>
<td>pmap_set[LSB]</td>
<td>wscat[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>chownat(GLIBC_2.4)</td>
<td>pmap_unset[LSB]</td>
<td>wscat[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fclose</td>
<td>poll[SUSv3]</td>
<td>wcscmp[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fcntl</td>
<td>popen[SUSv3]</td>
<td>wcscoll[LSB]</td>
<td></td>
</tr>
<tr>
<td>fcvt</td>
<td>posix_fadvise[SUSv3]</td>
<td>wcscpy[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fdatasync</td>
<td>posix_fadvise64[LSB]</td>
<td>wcscmpn[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fdoopen</td>
<td>posix_fallocate[SUSv3]</td>
<td>wcscdup[LSB]</td>
<td></td>
</tr>
<tr>
<td>fdopendir(GLIBC_2.4)</td>
<td>posix_fallocate64[LSB]</td>
<td>wctime[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>feof</td>
<td>posix_fadvise[SUSv3]</td>
<td>wcscpy[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>feof_unlocked</td>
<td>posix_memalign[SUSv3]</td>
<td>wcscnecmp[LSB]</td>
<td></td>
</tr>
<tr>
<td>ferror</td>
<td>posix_openpt[SUSv3]</td>
<td>wcscat[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>ferror_unlocked</td>
<td>posix_spawn[SUSv3]</td>
<td>wcscmp[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>execve</td>
<td>posix_spawn_file_action[SUSv3]</td>
<td>wcscat[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fflush</td>
<td>posix_spawn_file_action[SUSv3]</td>
<td>wcscpy[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fflush_unlocked</td>
<td>posix_spawn_file_action[SUSv3]</td>
<td>wcscpy[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fgetpos</td>
<td>posix_spawnattr_destroy[SUSv3]</td>
<td>wcsrchr[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fgetpos64</td>
<td>posix_spawnattr_destroy[SUSv3]</td>
<td>wcsrchr[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fgetpos</td>
<td>posix_spawnattr_getflags[SUSv3]</td>
<td>wcsspn[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>fgetpos64</td>
<td>posix_spawnattr_getpg</td>
<td>wcsstr[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgets[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgets_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetwc[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetwc_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetws[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetws_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fileno[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fileno_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flock[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flockfile[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fmemopen[SUSv4]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fnmatch[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fopen[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fopen64[LFS]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fork[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fpathconf[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fprintf[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fputs[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fputs_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fputswc[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fputswc_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fputws[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fputws_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fread[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fread_unlocked[LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>free[SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------</td>
<td>----------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>freeaddrinfo</td>
<td>SUSv3</td>
<td>raise</td>
<td>SUSv3</td>
</tr>
<tr>
<td>freopen</td>
<td>SUSv3</td>
<td>rand</td>
<td>SUSv3</td>
</tr>
<tr>
<td>freopen64</td>
<td>LFS</td>
<td>random</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fscanf</td>
<td>LSB</td>
<td>random_r</td>
<td>LSB</td>
</tr>
<tr>
<td>fseek</td>
<td>SUSv3</td>
<td>read</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fseeko</td>
<td>SUSv3</td>
<td>readdir</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fseek64</td>
<td>LFS</td>
<td>readdir64</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fseekpos</td>
<td>SUSv3</td>
<td>readdir64_r</td>
<td>LSB</td>
</tr>
<tr>
<td>fsetpos64</td>
<td>LFS</td>
<td>readdir_r</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fstatfs</td>
<td>LSB</td>
<td>readlink</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fstatfs64</td>
<td>LSB</td>
<td>readdirat</td>
<td>GLIBC_2.4</td>
</tr>
<tr>
<td>fstatvfs</td>
<td>SUSv3</td>
<td>readv</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fstatvfs64</td>
<td>LFS</td>
<td>realloc</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fsync</td>
<td>SUSv3</td>
<td>realpath</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftell</td>
<td>SUSv3</td>
<td>recv</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftello</td>
<td>SUSv3</td>
<td>recvfrom</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftello64</td>
<td>LFS</td>
<td>recvmsg</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftime</td>
<td>SUSv3</td>
<td>regcomp</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftok</td>
<td>SUSv3</td>
<td>regerror</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftruncate</td>
<td>SUSv3</td>
<td>regexec</td>
<td>LSB</td>
</tr>
<tr>
<td>ftruncate64</td>
<td>LFS</td>
<td>regfree</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fttylockfile</td>
<td>SUSv3</td>
<td>remove</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftw</td>
<td>SUSv3</td>
<td>remque</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftw64</td>
<td>LFS</td>
<td>rename</td>
<td>SUSv3</td>
</tr>
<tr>
<td>funlockfile</td>
<td>SUSv3</td>
<td>renameat</td>
<td>GLIBC_2.4</td>
</tr>
<tr>
<td>fwide</td>
<td>SUSv3</td>
<td>rewinding</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fwprintf</td>
<td>SUSv3</td>
<td>rewindingdir</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fwrite</td>
<td>SUSv3</td>
<td>rindex</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fwrite_unlocked</td>
<td>LSB</td>
<td>rmdir</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fwscanf</td>
<td>LSB</td>
<td>sbk</td>
<td>SUSv2</td>
</tr>
<tr>
<td>gai_strerror</td>
<td>SUSv3</td>
<td>scandir</td>
<td>SUSv4</td>
</tr>
<tr>
<td>gcvt</td>
<td>SUSv3</td>
<td>scandir64</td>
<td>LSB</td>
</tr>
<tr>
<td>getaddrinfo</td>
<td>SUSv3</td>
<td>scanf</td>
<td>LSB</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crypt</td>
<td>Encrypt block</td>
</tr>
<tr>
<td>encrypt</td>
<td>Encrypt block</td>
</tr>
<tr>
<td>setkey</td>
<td>Encrypt block</td>
</tr>
</tbody>
</table>

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>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dladdr</td>
<td>Dynamic link</td>
</tr>
<tr>
<td>dlerror</td>
<td>Dynamic link</td>
</tr>
<tr>
<td>dlclose</td>
<td>Dynamic link</td>
</tr>
<tr>
<td>dlsym</td>
<td>Dynamic link</td>
</tr>
</tbody>
</table>

A.4 libm

The behavior of the interfaces in this library is specified by the following Standards.

This Specification [LSB]
ISO POSIX (2003) [SUSv3]
SVID Issue 3 [SVID.3]

Table A-5 libm Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__finite</td>
<td>Finite value</td>
</tr>
<tr>
<td>__finitef</td>
<td>Finite value</td>
</tr>
<tr>
<td>__finitei</td>
<td>Finite value</td>
</tr>
<tr>
<td>_fpclassify</td>
<td>Floating point class</td>
</tr>
<tr>
<td>csinl</td>
<td>Sine value</td>
</tr>
<tr>
<td>csqrtl</td>
<td>Square root value</td>
</tr>
<tr>
<td>log</td>
<td>Logarithm value</td>
</tr>
<tr>
<td>llroundf</td>
<td>Long integer round</td>
</tr>
<tr>
<td>llroundl</td>
<td>Long integer round</td>
</tr>
<tr>
<td>log10</td>
<td>Common logarithm</td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>__fpclassifyf</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__signbitf</td>
<td>[LSB]</td>
</tr>
<tr>
<td>acos[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>acoshf[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>acosh[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>acos[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>acosl</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>asin[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>asinf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>asinh[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>asinhf</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>asinh[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>asin[USv3]</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>atanf2</td>
<td>[USv3]</td>
</tr>
<tr>
<td>atan2f</td>
<td>[USv3]</td>
</tr>
<tr>
<td>atanhf</td>
<td>[USv3]</td>
</tr>
<tr>
<td>atanh[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>atanh[USv3]</td>
<td>[USv3]</td>
</tr>
<tr>
<td>cargf</td>
<td>[USv3]</td>
</tr>
<tr>
<td>casinf</td>
<td>[USv3]</td>
</tr>
<tr>
<td>Function(s)</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>catanhf[SUSv3]</td>
<td>finite[LSB]</td>
</tr>
<tr>
<td>catanl[SUSv3]</td>
<td>finite[SUSv3]</td>
</tr>
<tr>
<td>cacos[SUSv3]</td>
<td>fesetexcept[SUSv3]</td>
</tr>
<tr>
<td>ccosp[SUSv3]</td>
<td>fesetround[SUSv3]</td>
</tr>
<tr>
<td>ccosh[SUSv3]</td>
<td>fesetexcept[SUSv3]</td>
</tr>
<tr>
<td>ccoshf[SUSv3]</td>
<td>fesetexceptflag[SUSv3]</td>
</tr>
<tr>
<td>ccoshl[SUSv3]</td>
<td>fesetround[SUSv3]</td>
</tr>
<tr>
<td>cexp[SUSv3]</td>
<td>finite[SUSv3]</td>
</tr>
<tr>
<td>cexpl[SUSv3]</td>
<td>finite[SUSv3]</td>
</tr>
<tr>
<td>clog10[SUSv3]</td>
<td>frexpl[SUSv3]</td>
</tr>
<tr>
<td>clog10f[SUSv3]</td>
<td>frexpl[SUSv3]</td>
</tr>
<tr>
<td>clog10l[SUSv3]</td>
<td>frexpl[SUSv3]</td>
</tr>
<tr>
<td>clog10[SUSv3]</td>
<td>frexpl[SUSv3]</td>
</tr>
<tr>
<td>clog10f[SUSv3]</td>
<td>frexpl[SUSv3]</td>
</tr>
<tr>
<td>clog10l[SUSv3]</td>
<td>frexpl[SUSv3]</td>
</tr>
<tr>
<td>copy(MSv3)</td>
<td>ilogbf[SUSv3]</td>
</tr>
<tr>
<td>copy(MSv3)</td>
<td>ilogbf[SUSv3]</td>
</tr>
<tr>
<td>copyf(MSv3)</td>
<td>j0[SUSv3]</td>
</tr>
<tr>
<td>copyf(MSv3)</td>
<td>j0f[LSB]</td>
</tr>
<tr>
<td>copyl(MSv3)</td>
<td>j1[LSB]</td>
</tr>
</tbody>
</table>
A.5 libncurses

The behavior of the interfaces in this library is specified by the following Standards.

This Specification [LSB]
X/Open Curses [SUS-CURSES]

Table A-7 libncurses Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>addch[SUS-CURSES]</td>
<td>mvdelch[SUS-CURSES]</td>
<td>slk_refresh[SUS-CURSES]</td>
</tr>
<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[LSB]</td>
<td>subpad[SUS-CURSES]</td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Function</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>attroff</td>
<td>[SUS-CURSES]</td>
<td>mvinnstr</td>
</tr>
<tr>
<td>attron</td>
<td>[SUS-CURSES]</td>
<td>mvinsch</td>
</tr>
<tr>
<td>attrset</td>
<td>[SUS-CURSES]</td>
<td>mvinsnstr</td>
</tr>
<tr>
<td>baudrate</td>
<td>[SUS-CURSES]</td>
<td>mvinsstr</td>
</tr>
<tr>
<td>beep</td>
<td>[SUS-CURSES]</td>
<td>mvprintw</td>
</tr>
<tr>
<td>bkgd</td>
<td>[SUS-CURSES]</td>
<td>mvprintw</td>
</tr>
<tr>
<td>bkgdset</td>
<td>[SUS-CURSES]</td>
<td>mvprintw</td>
</tr>
<tr>
<td>border</td>
<td>[SUS-CURSES]</td>
<td>mvwaddch</td>
</tr>
<tr>
<td>box</td>
<td>[SUS-CURSES]</td>
<td>mvwaddch</td>
</tr>
<tr>
<td>can_change_color</td>
<td>[SUS-CURSES]</td>
<td>mvwaddchnstr</td>
</tr>
<tr>
<td>cbreak</td>
<td>[SUS-CURSES]</td>
<td>mvwaddchstr</td>
</tr>
<tr>
<td>chgat</td>
<td>[SUS-CURSES]</td>
<td>mvwaddnstr</td>
</tr>
<tr>
<td>clear</td>
<td>[SUS-CURSES]</td>
<td>mvwaddstr</td>
</tr>
<tr>
<td>clearok</td>
<td>[SUS-CURSES]</td>
<td>mvwchgat</td>
</tr>
<tr>
<td>clrtobot</td>
<td>[SUS-CURSES]</td>
<td>mvwdelch</td>
</tr>
<tr>
<td>clrtoeol</td>
<td>[SUS-CURSES]</td>
<td>mvwgetch</td>
</tr>
<tr>
<td>color_content</td>
<td>[SUS-CURSES]</td>
<td>mvwgetnstr</td>
</tr>
<tr>
<td>color_set</td>
<td>[SUS-CURSES]</td>
<td>mvwgetstr</td>
</tr>
<tr>
<td>copywin</td>
<td>[SUS-CURSES]</td>
<td>mvwhline</td>
</tr>
<tr>
<td>curs_set</td>
<td>[SUS-CURSES]</td>
<td>mvwin</td>
</tr>
<tr>
<td>def_prog_mode</td>
<td>[SUS-CURSES]</td>
<td>mvwinch</td>
</tr>
<tr>
<td>def_shell_mode</td>
<td>[SUS-CURSES]</td>
<td>mvwinchnstr</td>
</tr>
<tr>
<td>del_curterm</td>
<td>[SUS-CURSES]</td>
<td>mvwinchstr</td>
</tr>
<tr>
<td>delay_output</td>
<td>[SUS-CURSES]</td>
<td>mvwinnstr</td>
</tr>
</tbody>
</table>

Annex A Alphabetical Listing of Interfaces
ISO/IEC 23360 Part 1:2008(E)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>delwin[SUS-CURSES]</td>
<td>mvwinsstr[LSB]</td>
<td>vwsanw[LSB]</td>
</tr>
<tr>
<td>douupdate[SUS-CURSES]</td>
<td>mvwscanw[LSB]</td>
<td>waddchnstr[SUS-CURSES]</td>
</tr>
<tr>
<td>echo[SUS-CURSES]</td>
<td>napms[SUS-CURSES]</td>
<td>waddnstr[SUS-CURSES]</td>
</tr>
<tr>
<td>getch[SUS-CURSES]</td>
<td>noqiflush[SUS-CURSES]</td>
<td>wbkgd[SUS-CURSES]</td>
</tr>
<tr>
<td>has_colors[SUS-CURSES]</td>
<td>pair_content[SUS-CURSES]</td>
<td>wclogbot[SUS-CURSES]</td>
</tr>
<tr>
<td>has_ic[SUS-CURSES]</td>
<td>pechochar[SUS-CURSES]</td>
<td>wclogteo[SUS-CURSES]</td>
</tr>
<tr>
<td>has_ill[SUS-CURSES]</td>
<td>pnoutrefresh[SUS-CURSES]</td>
<td>wcolor_set[SUS-CURSES]</td>
</tr>
</tbody>
</table>
An alphabetical listing of interfaces:

<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
<th>Function</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>idcok</td>
<td>SUS-CURSES</td>
<td>printw</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>idlok</td>
<td>SUS-CURSES</td>
<td>wdelch</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>immedok</td>
<td>SUS-CURSES</td>
<td>putp</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>indelk</td>
<td>SUS-CURSES</td>
<td>wdeleteml</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>inch</td>
<td>SUS-CURSES</td>
<td>qiflush</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>inchnstr</td>
<td>LSB</td>
<td>raw</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>inchstr</td>
<td>LSB</td>
<td>redrawwin</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>init_color</td>
<td>SUS-CURSES</td>
<td>refresh</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>init_pair</td>
<td>SUS-CURSES</td>
<td>reset_prog_mode</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>initscr</td>
<td>SUS-CURSES</td>
<td>reset_shell_mode</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>innstr</td>
<td>SUS-CURSES</td>
<td>ressetty</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>insch</td>
<td>SUS-CURSES</td>
<td>restartterm</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>insdelem</td>
<td>SUS-CURSES</td>
<td>ripofline</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>insertln</td>
<td>SUS-CURSES</td>
<td>savetty</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>insnstr</td>
<td>SUS-CURSES</td>
<td>scanw</td>
<td>LSB</td>
</tr>
<tr>
<td>insnstr</td>
<td>SUS-CURSES</td>
<td>scr_dump</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>instr</td>
<td>SUS-CURSES</td>
<td>scr_init</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>intrflush</td>
<td>SUS-CURSES</td>
<td>scr_restore</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>is_linetouched</td>
<td>SUS-CURSES</td>
<td>scr_set</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>is_wintouched</td>
<td>SUS-CURSES</td>
<td>scr</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>isendwin</td>
<td>SUS-CURSES</td>
<td>scroll</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>keyname</td>
<td>SUS-CURSES</td>
<td>scrollok</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>keypad</td>
<td>SUS-CURSES</td>
<td>set_curterm</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>killchar</td>
<td>SUS-CURSES</td>
<td>set_term</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>leaveok</td>
<td>SUS-CURSES</td>
<td>setscrreg</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>longname</td>
<td>SUS-CURSES</td>
<td>setupterm</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>meta</td>
<td>SUS-CURSES</td>
<td>slk_attr_set</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>meta</td>
<td>SUS-CURSES</td>
<td>wscroll</td>
<td>SUS-CURSES</td>
</tr>
<tr>
<td>meta</td>
<td>SUS-CURSES</td>
<td>wsetscrreg</td>
<td>SUS-CURSES</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
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>pamAuthenticate[ 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]

POSIX 1003.1 2008 [ SUSv4]
<table>
<thead>
<tr>
<th>Function Call</th>
<th>Library</th>
<th>Standard Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>_pthread_cleanup_pop</td>
<td>[LSB]</td>
<td></td>
</tr>
<tr>
<td>pthread_cond_timedwait</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_init</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>_pthread_cleanup_push</td>
<td>[LSB]</td>
<td></td>
</tr>
<tr>
<td>pthread_cond_wait</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_rdlock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>lseek64</td>
<td>[LFS]</td>
<td></td>
</tr>
<tr>
<td>pthread_condattr_destroy</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_timedlock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>open64</td>
<td>[LFS]</td>
<td></td>
</tr>
<tr>
<td>pthread_condattr_getpshared</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_timedlock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pread</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_condattr_init</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_trylock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pread64</td>
<td>[LSB]</td>
<td></td>
</tr>
<tr>
<td>pthread_condattr_setpshared</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_trylock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_destroy</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_create</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_unlock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getdetachstate</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_detach</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_wlock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getguardsize</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_equal</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_destroy</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getinheritstate</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_exit</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_getpshared</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getconcurrency</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_init</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getcpuclockid</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_setpshared</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getscope</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getschedparam</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_self</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstack</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getspecific</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setschedparam</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstackaddr</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_join</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setcancelstate</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_getstacksize</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_key_create</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setconcurrency</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_init</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_key_delete</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setschedparam</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setdetachstate</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_kill</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setschedprio</td>
<td>[GLIBC_2.3.4]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setseguardsize</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_destroy</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_setpshared</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_getprioceiling</td>
<td>[GLIBC_2.4]</td>
<td></td>
</tr>
<tr>
<td>pthread_spin_destroy</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setschedparam</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setseguardsize</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_init</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setcancelstate</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_lock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setcanceltype</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_unlock</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_equal</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_setpshared</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_attr_setscheduler</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>SUSv3</td>
<td>SUSv4</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>pthread_attr_setscope</td>
<td>pthread_mutex_setpriority ceiling(GLIBC_2.4)</td>
<td>pthread_spin_lock</td>
</tr>
<tr>
<td>pthread_attr_setstack</td>
<td>pthread_mutex_timedlock</td>
<td>pthread_spin_trylock</td>
</tr>
<tr>
<td>pthread_attr_setstackaddr</td>
<td>pthread_mutex_trylock</td>
<td>pthread_spin_unlock</td>
</tr>
<tr>
<td>pthread_attr_setstacksize</td>
<td>pthread_mutex_unlock</td>
<td>pthread_mutex_unlock</td>
</tr>
<tr>
<td>pthread_barrier_destroy</td>
<td>pthread_mutexattr_destroy</td>
<td>pthread_mutexattr_destroy</td>
</tr>
<tr>
<td>pthread_barrier_init</td>
<td>pthread_mutexattr_getpriority ceiling(GLIBC_2.4)</td>
<td>pthread_mutexattr_getprotocol(GLIBC_2.4)</td>
</tr>
<tr>
<td>pthread_barrier_wait</td>
<td>pthread_mutexattr_getprotocol(GLIBC_2.4)</td>
<td>pthread_mutexattr_getpshared</td>
</tr>
<tr>
<td>pthread_barrierattr_destroy</td>
<td>pthread_mutexattr_getpshared</td>
<td>pthread_mutexattr_getpshared</td>
</tr>
<tr>
<td>pthread_barrierattr_getpshared(GLIBC_2.3.3)</td>
<td>pthread_mutexattr_getpshared</td>
<td>pthread_mutexattr_getpshared</td>
</tr>
<tr>
<td>pthread_barrierattr_init</td>
<td>pthread_mutexattr_init</td>
<td>pthread_mutexattr_init</td>
</tr>
<tr>
<td>pthread_barrierattr_setpshared(GLIBC_2.3.3)</td>
<td>pthread_mutexattr_setpshared</td>
<td>pthread_mutexattr_setpshared</td>
</tr>
<tr>
<td>pthread_cancel</td>
<td>pthread_mutexattr_setpriority ceiling(GLIBC_2.4)</td>
<td>pthread_mutexattr_setprotocol(GLIBC_2.4)</td>
</tr>
<tr>
<td>pthread_cond_broadcast</td>
<td>pthread_mutexattr_setpshared</td>
<td>pthread_mutexattr_setpshared</td>
</tr>
<tr>
<td>pthread_cond_destroy</td>
<td>pthread_mutexattr_setpshared</td>
<td>pthread_mutexattr_setpshared</td>
</tr>
<tr>
<td>pthread_cond_init(GLIBC_2.3.3)</td>
<td>pthread_mutexattr_setpshared</td>
<td>pthread_mutexattr_setpshared</td>
</tr>
<tr>
<td>pthread_cond_signal</td>
<td>pthread_rwlock_destroy</td>
<td>pthread_rwlock_destroy</td>
</tr>
<tr>
<td>clock_getcpu_clockid</td>
<td>mq_open(GLIBC_2.3.4)</td>
<td>shm_unlink</td>
</tr>
</tbody>
</table>

### 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>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock_getcpu_clockid</td>
<td>mq_open(GLIBC_2.3.4)</td>
<td>shm_unlink</td>
<td>shm_unlink</td>
</tr>
</tbody>
</table>
### Annex A Alphabetical Listing of Interfaces

#### ISO/IEC 23360 Part 1:2008(E)

<table>
<thead>
<tr>
<th>Function</th>
<th>Specifies</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock_getres[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock_gettime[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock_nanosleep[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clock_settime[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_receive(GLIBC_2.3.4)</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>mq_send(GLIBC_2.3.4)</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>mq_setattr(GLIBC_2.3.4)[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_timedreceive(GLIBC_2.3.4)[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_timedsend(GLIBC_2.3.4)[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_unlink(GLIBC_2.3.4)[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mq_notify(GLIBC_2.3.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shm_open[SUSv3]</td>
<td></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>Specifies</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>forkpty[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>login_tty[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logwtmp[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>login[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logout[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>openpty[LSB]</td>
<td></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>Specifies</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>adler32[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzclose[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gztell[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compress[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzdopen[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzwrite[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compress2[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzeof[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflate[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>compressBound[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzerror[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateEnd[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>crc32[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzfllush[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateInit2_[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflate[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzgetc[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateInit_[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateBound[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzipets[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateReset[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateCopy[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzopen[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateSetDictionary[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateEnd[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzprintf[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateSync[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateInit2_[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzputc[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inflateSyncPoint[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateInit_[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzputs[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uncompress[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateParams[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzread[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zError[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateReset[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzrewind[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zlibVersion[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>deflateSetDictionary[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzseek[LSB]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>B</code></td>
<td><code>get_crc_table[LSB]</code></td>
<td><code>gzsetparams[LSB]</code></td>
</tr>
<tr>
<td>-----</td>
<td>----------------------</td>
<td>--------------------</td>
</tr>
</tbody>
</table>

© 2008 Linux Foundation
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 `--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 `--remove` be required to be the exact reverse of installation.

### Object Types

The `-t` `type` or `--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 `.sh`.

The `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 `-I` (the letter *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 `%d/%s` with the port number and protocol values (e.g. `22/tcp`), and the second operand should be the name of the service. Any subsequent operands provide aliases for this service. The `-p pkg` or `--package=pkg` option is not required for service objects, and is ignored if specified. If any of the `-r`, `--remove`, `-c` or `--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 `-r` or `--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 `-r`, `--remove`, `-c` or `--check` options are specified, the first operand should have the format:

"%s:%s:%s:%s:%s:%s"

Otherwise, the first operand should have the format

"%s:%s"

The fields in the first operand have the following meaning, in order:

svc_name
The name of this service. If the name does not contain a /, this should match the name of an already installed service (see also getservbyname()). If the name contains a / 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**

```
lsbinstall --package=myapp --type=profile myco.com-prod.sh
```

Install the profile shell script for myco.com-prod.sh, part of the myapp package.

```
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. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

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 retile 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 ti-
tle 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.