INTERNATIONAL STANDARD

ISO/IEC 23360-1-2

First edition
2021-10

Linux Standard Base (LSB) —

Part 1-2:
Core specification generic part

Reference number
ISO/IEC 23360-1-2:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the Linux Foundation as Linux Standard Base (LSB): Core specification generic part and drafted in accordance with its editorial rules. It was assigned to Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*, and adopted by National Bodies.

This first edition of ISO/IEC 23360-1-2 cancels and replaces ISO/IEC 23360-1:2006, which has been technically revised.

This document is based on "The GNU Free Documentation License, version 1.1". The license is available at https://www.gnu.org/licenses/old-licenses/fdl-1.1.html.

A list of all parts in the ISO/IEC 23660 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Contents

Introduction

The LSB defines a binary interface for application programs that are compiled and packaged for LSB-conforming implementations on many different hardware architectures. A binary specification must include information specific to the computer processor architecture for which it is intended. To avoid the complexity of conditional descriptions, the specification has instead been divided into generic parts which are augmented by one of several architecture-specific parts, depending on the target processor architecture; the generic part will indicate when reference must be made to the architecture part, and vice versa.

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

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

1. The first number (`x`) is the major version number. Versions sharing the same major version number shall be compatible in a backwards direction; that is, a newer version shall be compatible with an older version. Any deletion of a library results in a new major version number. Interfaces marked as deprecated may be removed from the specification at a major version change.
2. The second number (`y`) is the minor version number. Libraries and individual interfaces may be added, but not removed. Interfaces may be marked as deprecated at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.
3. The third number (`z`), if present, is the editorial level. Only editorial changes should be included in such versions.

Since this specification is a descriptive Application Binary Interface, and not a source level API specification, it is not possible to make a guarantee of 100% backward compatibility between major releases. However, it is the intent that those parts of the binary interface that are visible in the source level API will remain backward compatible from version to version, except where a feature marked as "Deprecated" in one release may be removed from a future release. Implementors are strongly encouraged to make use of symbol versioning to permit simultaneous support of applications conforming to different releases of this specification.

LSB is a trademark of the Linux Foundation. Developers of applications or implementations interested in using the trademark should see the Linux Foundation Certification Policy for details.

I Introductory Elements

1 Scope

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 part describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific part describing the parts of the interface that vary by processor architecture. Together, the common part and the relevant architecture-specific part for a single hardware architecture provide a complete interface specification for compiled application programs on systems that share a common hardware architecture.

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.

This is the common part of the Core module of the Linux Standard Base (LSB), LSB Core - Generic. This module provides the fundamental system interfaces, libraries, and runtime environment upon which all conforming applications and libraries depend.

LSB Core - Generic, the common part, should be used in conjunction with an architecture-specific part. Whenever a section of the common part is supplemented by architecture-specific information, the common part includes a reference to the architecture-specific part. Architecture-specific parts of the LSB Core Specification may also contain additional information that is not referenced in the common part.

Interfaces described in this part of the LSB Core Specification are mandatory except where explicitly listed otherwise. Interfaces described in the LSB Core module are supplemented by other LSB modules. All other modules depend on the presence of LSB Core.

2 References

2.1 Normative References

The following specifications are incorporated by reference into this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced specification (including any amendments) applies.

Note: Where copies of a referenced specification are available on the World Wide Web, a Uniform Resource Locator (URL) is given, for informative purposes only. Such URL might at any given time resolve to a more recent copy of the specification, or be out of date (not resolve). Reference copies of specifications at the revision level indicated may be found at the Linux Foundation's Reference Specifications (http://refspecs.linuxbase.org) site.

Table 2-1 Normative References

Name	Title	URL
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 3.0	http://refspecs.linuxbase.org/fhs
ISO C (1999)	ISO/IEC 9899:1999 - Programming Languages -- C	
ISO/IEC 14882: 2003 C++ Language	ISO/IEC 14882: 2003 Programming languages -- C++	
Itanium™ C++ ABI	Itanium™ C++ ABI (Revision 1.86)	http://refspecs.linuxfoundation.org/cxxabi-1.86.html
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
Libncursesw API	Libncursesw API	http://invisible-island.net/ncurses/man/ncurses.3x.html
Libncursesw Placeholder	Libncursesw Specification Placeholder	http://refspecs.linux-foundation.org/libncursesw/libncurses.html
POSIX 1003.1-2001 (ISO/IEC 9945-2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface	http://www.unix.org/version3/

Name	Title	URL
	(POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004	
POSIX 1003.1-2008 (ISO/IEC 9945-2009)	Portable Operating System Interface (POSIX®) 2008 Edition / The Open Group Technical Standard Base Specifications, Issue 7	http://www.unix.org/version4/
SUSv2	CAE Specification, January 1997, System Interfaces and Headers (XSH),Issue 5 (ISBN: 1-85912-181-0, C606)	http://www.opengroup.org/publications/catalog/un.htm
SVID Issue 3	American Telephone and Telegraph Company, System V Interface Definition, Issue 3; Morristown, NJ, UNIX Press, 1989. (ISBN 0201566524)	
SVID Issue 4	System V Interface Definition, Fourth Edition	http://refspecs.linuxfoundation.org/svid4/
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.sco.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface -	http://www.sco.com/developers/gabi/2003-12-17/contents.html

Name	Title	URL
	DRAFT - 17 December 2003	
X/Open Curses, Issue 7	X/Open Curses, Issue 7 (ISBN: 1-931624-83-6, The Open Group, November 2009)	https://www2.opengroup.org/ogsys/catalog/C094

2.2 Informative References/Bibliography

The documents listed below provide essential background information to implementors of this specification. These references are included for information only, and do not represent normative parts of this specification.

Table 2-2 Other References

Name	Title	URL
DWARF Debugging Information Format, Version 4	DWARF Debugging Information Format, Version 4 (June 10, 2010)	http://www.dwarfstd.org/doc/DWARF4.pdf
IEC 60559/IEEE 754 Floating Point	IEC 60559:1989 Binary floating-point arithmetic for microprocessor systems	http://www.ieee.org/
ISO/IEC TR14652	ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions	
ITU-T V.42	International Telecommunication Union Recommendation V.42 (2002): Error-correcting procedures for DCEs using asynchronous-to-synchronous conversionITUV	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Li18nux Globalization Specification	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.openi18n.org/docs/html/LI18NUX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices-2.6+.txt

Name	Title	URL
Linux Assigned Names And Numbers Authority	Linux Assigned Names And Numbers Authority	http://www.lanana.org/
Mozilla's NSS SSL Reference	Mozilla's NSS SSL Reference	http://www.mozilla.org/projects/security/pki/nss/ref/ssl/
NSPR Reference	Mozilla's NSPR Reference	http://refspecs.linuxfoundation.org/NSPR_API_Reference/NSPR_API.html
PAM	Open Software Foundation, Request For Comments: 86.0 , October 1995, V. Samar & R.Schemers (SunSoft)	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1950: ZLIB Compressed Data Format Specication	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821:Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822:Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 5531/4506 RPC & XDR	IETF RFC 5531 & 4506	http://www.ietf.org/

Name	Title	URL
RFC 791:Internet Protocol	IETF RFC 791: Internet Protocol Specification	http://www.ietf.org/rfc/rfc791.txt
RPM Package Format	RPM Package Format V3.0	http://www.rpm.org/max-rpm/s1-rpm-file-format-rpm-file-format.html
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

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 the LSB Core Specification.

Table 3-1 Standard Library Names

Library	Runtime Name
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libgcc_s	libgcc_s.so.1
libncurses	libncurses.so.5
libncursesw	libncursesw.so.5
libnspr4	libnspr4.so
libnss3	libnss3.so
libpam	libpam.so.0
libpthread	libpthread.so.0
librt	librt.so.1
libssl3	libssl3.so
libstdcxx	libstdc++.so.6
libutil	libutil.so.1
libz	libz.so.1

Table 3-2 Standard Library Names defined in the Architecture Specific Parts of the LSB Core Specification

Library	Runtime Name
libc	See architecture specific part.
libm	See architecture specific part.
proginterp	See architecture specific part.

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 (LSB

Core - Generic) and the relevant architecture specific part of the LSB Core Specification.

> **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 the LSB Core Specification that supplement this specification for a given target processor architecture describe a minimum acceptable processor. The implementation shall provide all features of this processor, whether in hardware or through emulation transparent to the application.
- The implementation shall be capable of executing compiled applications having the format and using the system interfaces described in this specification.
- The implementation shall provide libraries containing the interfaces specified by this specification, 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 specification.
- The map of virtual memory provided by the implementation shall conform to the requirements of this specification.
- 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 specification.
- 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 specification in the format defined here and in other documents normatively included by reference. All commands and utilities shall behave as required by this specification. The implementation shall also provide all mandatory components of an application's runtime environment that are included or referenced in this specification.
- 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 containing object files is necessarily architecture specific, and must conform to both the generic LSB Core specification (LSB Core - Generic) and the relevant architecture specific part of the LSB Core Specification. A conforming application which contains no object files may be architecture neutral. Architecture neutral applications shall conform only to the requirements of the generic LSB Core specification (LSB Core - Generic).

A conforming application shall satisfy the following requirements:

- Executable files shall be either object files in the format defined in the Object Format section of this specification, or script files in a scripting language where the interpreter is required by this specification.
- Object files shall participate in dynamic linking as defined in the Program Loading and Linking section of this specification.
- Object files shall employ only the instructions, traps, and other low-level facilities defined as being for use by applications in the Low-Level System Information section of this specification
- If the application requires any optional interface defined in this specification in order to be installed or to execute successfully, the requirement for that optional interface shall be stated in the application's documentation.
- The application shall not use any interface or data format that is not required to be provided by a conforming implementation, unless such an interface or data format is supplied by another application through direct invocation of that application during execution. The other application must also be a conforming application, and the use of such interface or data format, as well as its source (in other words, the other conforming application), shall be identified in the documentation of the application.
- The application 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 not defined in this specification in order to be installed or to execute successfully.

4 Terms and Definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 2382, ISO 80000–2, and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

4.1

archLSB

Some LSB specification documents have both a generic, architecture-neutral part and an architecture-specific part. The latter describes elements whose definitions may be unique to a particular processor architecture. The term archLSB may be used in the generic part to refer to the corresponding section of the architecture-specific part.

4.2

Binary Standard, ABI

The total set of interfaces that are available to be used in the compiled binary code of a conforming application, including the run-time details such as calling conventions, binary format, C++ name mangling, etc.

4.3

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.

4.4

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.

4.5

Source Standard, API

The total set of interfaces that are available to be used in the source code of a conforming application. Due to translations, the Binary Standard and the Source Standard may contain some different interfaces.

4.6

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.

4.7

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.

In addition, for the portions of this specification which build on IEEE Std 1003.1-2001, the definitions given in *IEEE Std 1003.1-2001, Base Definitions, Chapter 3* apply.

5 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) [SUSv4]

refers to the interface named `forkpty()` with symbol version `GLIBC_2.0` that is defined in the reference indicated by the tag `SUSv4`.

Note: For symbols with versions which differ between architectures, the symbol versions are defined in the architecture specific parts of of this module specification only. In the generic part, they will appear without symbol versions.

6 Relationship To ISO/IEC 9945 POSIX

This specification includes many interfaces described in POSIX 1003.1-2008 (ISO/IEC 9945-2009). Unless otherwise specified, such interfaces should behave exactly as described in that specification. Any conflict between the requirements described here and the POSIX 1003.1-2008 (ISO/IEC 9945-2009) standard is unintentional, except as explicitly noted otherwise.

> **Note:** In addition to the differences noted in this specification, a report, *ISO/IEC TR 24715-Technical Report on the Conflicts Between the ISO/IEC 9945 (POSIX) Standard and the Linux Standard Base Specification (LSB)*, identifies the differences between edition 3.1 of this specification and *POSIX 1003.1-2001 (ISO/IEC 9945-2003)* (more precisely, POSIX 2001 plus the first two corrigenda, informally known as the 2004 edition). It is the long term plan of the Linux Foundation to converge the LSB Core specification with the ISO/IEC POSIX specification.

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.

7 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.

ISO/IEC 23360 corresponds to an earlier edition of this specification (version 3.1), published as an ISO/IEC standard in 2006 after submission by the Linux Foundation. The ISO edition is also the subject of the technical report ISO/IEC TR 24715 referenced in the previous chapter.

II Executable And Linking Format (ELF)

8 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.

9 Low Level System Information

9.1 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.

9.2 Machine Interface

9.2.1 Data Representation

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

9.2.1.1 Fundamental Types

In addition to the fundamental types specified in the relevant architecture specific part of the LSB Core Specification, a 1 byte data type is defined here.

Table 9-1 Scalar Types

Type	C	C++	`sizeof`	Align-ment (bytes)	Architec-ture Rep-resenta-tion
Integral	_Bool	bool	1	1	byte

10 Object Format

10.1 Object Files

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

- System V ABI
- System V ABI Update
- the relevant architecture specific ABI supplement.
- this specification
- the relevant architecture specific part of the LSB Core Specification

Conforming implementations may also support other unspecified object file formats.

10.2 Sections

10.2.1 Introduction

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

10.2.2 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 10-1, drawn from the System V ABI, or one of the additional values specified in Table 10-2.

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

10.2.2.1 ELF Section Types

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

Table 10-1 ELF Section Types

Name	Value	Description
SHT_DYNAMIC	0x6	The section holds information for dynamic linking. Currently, an object file shall have only one dynamic section, but this restriction may be relaxed in the future. See `Dynamic Section' in Chapter 5 of System V ABI Update for details.
SHT_DYNSYM	0xb	This section holds a minimal set of symbols adequate for dynamic

Name	Value	Description
		linking. See also SHT_SYMTAB. Currently, an object file may have either a section of SHT_SYMTAB type or a section of SHT_DYNSYM type, but not both. This restriction may be relaxed in the future.
SHT_FINI_ARRAY	0xf	This section contains an array of pointers to termination functions, as described in `Initialization and Termination Functions' in Chapter 5 of System V ABI Update. Each pointer in the array is taken as a parameterless procedure with a void return.
SHT_HASH	0x5	The section holds a symbol hash table. Currently, an object file shall have only one hash table, but this restriction may be relaxed in the future. See `Hash Table' in Chapter 5 of System V ABI Update for details.
SHT_INIT_ARRAY	0xe	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.
SHT_NOBITS	0x8	A section of this type occupies no space in the file but otherwise re-

Name	Value	Description
		sembles SHT_PROG-BITS. Although this section contains no bytes, the sh_offset member contains the conceptual file offset.
SHT_NOTE	0x7	The section holds information that marks the file in some way. See `Note Section' in Chapter 5 of System V ABI Update for details.
SHT_NULL	0x0	This value marks the section header as inactive; it does not have an associated section. Other members of the section header have undefined values.
SHT_PREINIT_ARRAY	0x10	This section contains an array of pointers to functions that are invoked before all other initialization functions, as described in `Initialization and Termination Functions' in Chapter 5 of System V ABI Update. Each pointer in the array is taken as a parameterless proceure with a void return.
SHT_PROGBITS	0x1	The section holds information defined by the program, whose format and meaning are determined solely by the program.
SHT_REL	0x9	The section holds relocation entries without explicit addends, such as type Elf32_Rel for the 32-bit class of object files or type Elf64_Rel for the 64-bit class of object files. An object

Name	Value	Description
		file may have multiple relocation sections. See `Relocation' in Chapter 4 of System V ABI Update for details.
SHT_RELA	0x4	The section holds relocation entries with explicit addends, such as type Elf32_Rela for the 32-bit class of object files or type Elf64_Rela for the 64-bit class of object files. An object file may have multiple relocation sections. 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.

10.2.2.2 Additional Section Types

The following additional section types are defined here.

Table 10-2 Additional Section Types

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

10.3 Special Sections

10.3.1 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.

10.3.1.1 ELF Special Sections

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

Table 10-3 ELF Special Sections

Name	Type	Attributes
.bss	SHT_NOBITS	SHF_ALLOC+SHF_WRITE
.comment	SHT_PROGBITS	SHF_MERGE+SHF_STRINGS
.data	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
.data1	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
.debug	SHT_PROGBITS	0
.dynamic	SHT_DYNAMIC	SHF_ALLOC+SHF_WRITE
.dynstr	SHT_STRTAB	SHF_ALLOC
.dynsym	SHT_DYNSYM	SHF_ALLOC
.fini	SHT_PROGBITS	SHF_ALLOC+SHF_EXECINSTR
.fini_array	SHT_FINI_ARRAY	SHF_ALLOC+SHF_WRITE

Name	Type	Attributes
.hash	SHT_HASH	SHF_ALLOC
.init	SHT_PROGBITS	SHF_ALLOC+SHF_EX-ECINSTR
.init_array	SHT_INIT_ARRAY	SHF_AL-LOC+SHF_WRITE
.interp	SHT_PROGBITS	SHF_ALLOC
.line	SHT_PROGBITS	0
.note	SHT_NOTE	0
.preinit_array	SHT_PREINIT_ARRAY	SHF_AL-LOC+SHF_WRITE
.rodata	SHT_PROGBITS	SHF_AL-LOC+SHF_MERGE+SHF_STRINGS
.rodata1	SHT_PROGBITS	SHF_AL-LOC+SHF_MERGE+SHF_STRINGS
.shstrtab	SHT_STRTAB	0
.strtab	SHT_STRTAB	SHF_ALLOC
.symtab	SHT_SYMTAB	SHF_ALLOC
.tbss	SHT_NOBITS	SHF_AL-LOC+SHF_WRITE+SHF_TLS
.tdata	SHT_PROGBITS	SHF_AL-LOC+SHF_WRITE+SHF_TLS
.text	SHT_PROGBITS	SHF_ALLOC+SHF_EX-ECINSTR

.bss

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

.comment

This section holds version control information.

.data

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

.data1

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

.debug

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

.dynamic

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

.dynstr

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

.dynsym

This section holds the dynamic linking symbol table, as described in `Symbol Table' of System V ABI Update.

.fini

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

.fini_array

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

.hash

This section holds a symbol hash table. See `Hash Table' in Chapter 5 of System V ABI Update for more information.

.init

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

.init_array

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

.interp

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

.line

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

.note

This section holds information in the format that `Note Section' in Chapter 5 of System V ABI Update describes.

.preinit_array

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

.rodata

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

.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_ALLOC bit; otherwise, that bit will be off.

.symtab

This section holds a symbol table, as `Symbol Table' in Chapter 4 of System V ABI Update describes. If the file has a loadable segment that includes the symbol table, the section's attributes will include the SHF_ALLOC bit; otherwise, that bit will be off.

.tbss

This section holds uninitialized thread-local data that contribute to the program's memory image. By definition, the system initializes the data with zeros when the data is instantiated for each new execution flow. The section

occupies no file space, as indicated by the section type, SHT_NOBITS. Implementations need not support thread-local storage.

.tdata

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

.text

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

10.3.1.2 Additional Special Sections

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

Table 10-4 Additional Special Sections

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

.ctors

This section contains a list of global constructor function pointers.

.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 10.6.1.

.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 10.6.2.

.gcc_except_table

This section holds Language Specific Data.

.gnu.version

This section contains the Symbol Version Table. See Section 10.7.2.

.gnu.version_d

This section contains the Version Definitions. See Section 10.7.3.

.gnu.version_r

This section contains the Version Requirements. See Section 10.7.4.

.got.plt

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

.jcr

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

.note.ABI-tag

Specify ABI details. See Section 10.8.

.stab

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

.stabstr

This section contains strings associated with the debugging infomation contained in the .stab section.

10.4 Symbol Mapping

10.4.1 Introduction

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

10.4.1.1 C Language

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

10.5 DWARF Extensions

The LSB does not specify debugging information, however, some additional sections contain information which is encoded using the the encoding as specified by DWARF Debugging Information Format, Version 4 with extensions defined here.

10.5.1 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 10-5 DWARF Exception Header value format

Name	Value	Meaning
DW_EH_PE_absptr	0x00	The Value is a literal pointer whose size is determined by the architecture.
DW_EH_PE_uleb128	0x01	Unsigned value is encoded using the Little Endian Base 128 (LEB128) as defined by DWARF Debugging Information Format, Version 4.
DW_EH_PE_udata2	0x02	A 2 bytes unsigned value.
DW_EH_PE_udata4	0x03	A 4 bytes unsigned value.
DW_EH_PE_udata8	0x04	An 8 bytes unsigned value.
DW_EH_PE_sleb128	0x09	Signed value is encoded using the Little Endian

Name	Value	Meaning
		Base 128 (LEB128) as defined by DWARF Debugging Information Format, Version 4.
DW_EH_PE_sdata2	0x0A	A 2 bytes signed value.
DW_EH_PE_sdata4	0x0B	A 4 bytes signed value.
DW_EH_PE_sdata8	0x0C	An 8 bytes signed value.

Table 10-6 DWARF Exception Header application

Name	Value	Meaning
DW_EH_PE_pcrel	0x10	Value is relative to the current program counter.
DW_EH_PE_textrel	0x20	Value is relative to the beginning of the .text section.
DW_EH_PE_datarel	0x30	Value is relative to the beginning of the .got or .eh_frame_hdr section.
DW_EH_PE_funcrel	0x40	Value is relative to the beginning of the function.
DW_EH_PE_aligned	0x50	Value is aligned to an address unit sized boundary.

One special encoding, 0xff (DW_EH_PE_omit), shall be used to indicate that no value ispresent.

10.5.2 DWARF CFI Extensions

In addition to the Call Frame Instructions defined in section 6.4.2 of DWARF Debugging Information Format, Version 4, the following additional Call Frame Instructions may also be used.

Table 10-7 Additional DWARF Call Frame Instructions

Name	Value	Meaning
DW_CFA_GNU_args_size	0x2e	The DW_CFA_GNU_args_size instruction takes an unsigned LEB128 operand representing

Name	Value	Meaning
		an argument size. This instruction specifies the total of the size of the arguments which have been pushed onto the stack.
DW_CFA_GNU_negative_offset_extended	0x2f	The DW_CFA_def_cfa_sf instruction takes two operands: an unsigned LEB128 value representing a register number and an unsigned LEB128 which represents the magnitude of the offset. This instruction is identical to DW_CFA_offset_extended_sf except that the operand is subtracted to produce the offset. This instructions is obsoleted by DW_CFA_offset_extended_sf.

10.6 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, Version 4. Readers are advised that there are some subtle difference, and care should be taken when comparing the two sections.

10.6.1 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 10-8 Call Frame Information Format

Common Information Entry Record
Frame Description Entry Record(s)

10.6.1.1 The Common Information Entry Format

Table 10-9 Common Information Entry Format

Length	Required
Extended Length	Optional
CIE ID	Required
Version	Required
Augmentation String	Required
Code Alignment Factor	Required
Data Alignment Factor	Required
Return Address Register	Required
Augmentation Data Length	Optional
Augmentation Data	Optional
Initial Instructions	Required
Padding	

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 themselves. This field is not present unless the *Length* field contains the value 0xffffffff.

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 adavance 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 Augentation 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.

10.6.1.1.1 Augmentation String Format

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

'z'

A 'z' may be present as the first character of the string. If present, the Augmentation Data field shall be present. The contents of the Augmentation Data shall be intepreted 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.

10.6.1.2 The Frame Description Entry Format

Table 10-10 Frame Description Entry Format

Length	Required
Extended Length	Optional
FDE Pointer	Required
PC Begin	Required
PC Range	Required
Augmentation Data Length	Optional
Augmentation Data	Optional
Call Frame Instructions	Required
Padding	

`Length`

A 4 byte unsigned value indicating the length in bytes of the FDE 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 FDE shall be considered a terminator and processing shall end.

Extended Length

A 8 byte unsigned value indicating the length in bytes of the FDE structure, not including the *Length* or *Extended Length* field themselves. This field is not present unless the *Length* field contains the value 0xffffffff.

CIE Pointer

A 4 byte unsigned value that when subtracted from the offset of 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 Augentation Length.

Call Frame Instructions

A set of Call Frame Instructions.

Padding

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

10.6.2 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 10.5.1.

Table 10-11 `.eh_frame_hdr` Section Format

Encoding	Field
unsigned byte	version
unsigned byte	eh_frame_ptr_enc
unsigned byte	fde_count_enc

Encoding	Field
unsigned byte	table_enc
encoded	eh_frame_ptr
encoded	fde_count
	binary search table

version

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

eh_frame_ptr_enc

The encoding format of the eh_frame_ptr field.

fde_count_enc

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

table_enc

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

eh_frame_ptr

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

fde_count

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

binary search table

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

10.7 Symbol Versioning

10.7.1 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.

10.7.2 Symbol Version Table

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

The `.gnu.version` section shall contain an array of elements of type `Elfxx_Half`. Each entry specifies the version defined for or required by the corresponding symbol in the Dynamic Symbol Table.

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

The values 0 and 1 are reserved.

0

The symbol is local, not available outside the object.

1

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

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

10.7.3 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 10-1, optionally followed by an array of `Elfxx_Verdaux` structures, as defined in Figure 10-2.

```
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 10-1 Version Definition Entries

`vd_version`

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

`vd_flags`

Version information flag bitmask.

vd_ndx

Version index numeric value referencing the SHT_GNU_versym section.

vd_cnt

Number of associated verdaux array entries.

vd_hash

Version name hash value (ELF hash function).

vd_aux

Offset in bytes to a corresponding entry in an array of `Elfxx_Verdaux` structures as defined in Figure 10-2

vd_next

Offset to the next verdef entry, in bytes.

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

Figure 10-2 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.

10.7.4 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 10-3, optionally followed by an array of `Elfxx_Vernaux` structures, as defined in Figure 10-4.

```
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 10-3 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 10-4 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.

10.7.5 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.

10.7.6 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 symbolis 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.

10.8 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.

11 Dynamic Linking

11.1 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 the LSB Core Specification.

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

11.2 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 11-1 Linux Segment Types

Name	Value
PT_GNU_EH_FRAME	0x6474e550
PT_GNU_STACK	0x6474e551
PT_GNU_RELRO	0x6474e552

PT_GNU_EH_FRAME

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

PT_GNU_STACK

The *p_flags* member specifies the permissions on the segment containing the stack and is used to indicate wether the stack should be executable. The absense of this header indicates 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.

11.3 Dynamic Entries

11.3.1 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.

```
typedef struct {
       Elf32_Sword    d_tag;
       union {
              Elf32_Word     d_val;
              Elf32_Addr     d_ptr;
       } d_un;
} Elf32_Dyn;
```

```
extern Elf32_Dyn        _DYNAMIC[];

typedef struct {
        Elf64_Sxword    d_tag;
        union {
                Elf64_Xword     d_val;
                Elf64_Addr      d_ptr;
        } d_un;
} Elf64_Dyn;

extern Elf64_Dyn        _DYNAMIC[];
```

Figure 11-1 Dynamic Structure

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

11.3.2 Dynamic Entries

11.3.2.1 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_FINI_ARRAY

The address of an array of pointers to termination functions.

DT_FINI_ARRAYSZ

Size in bytes of DT_FINI_ARRAY

DT_FLAGS

Flag values specific to the object being loaded

DT_HASH

Address of symbol hash table

DT_HIPROC

End of processor-specific

DT_INIT

Address of init function

DT_INIT_ARRAY

The address of an array of pointers to initialization functions.

DT_INIT_ARRAYSZ

Size in bytes of DT_INIT_ARRAY

DT_JMPREL

Address of PLT relocs

DT_LOPROC

Start of processor-specific

DT_NEEDED

Name of needed library

DT_NULL

Marks end of dynamic section

DT_PLTREL

Type of reloc in PLT

DT_PLTRELSZ

Size in bytes of PLT relocs

DT_PREINIT_ARRAY

Array with addresses of preinit functions

DT_PREINIT_ARRAYSZ

Size in bytes of DT_PREINIT_ARRAY

DT_REL

Address of Rel relocs

DT_RELA

Address of Rela relocs

DT_RELAENT

Size of one Rela reloc

DT_RELASZ

Total size of Rela relocs

DT_RELENT

Size of one Rel reloc

DT_RELSZ

Total size of Rel relocs

DT_RPATH

Library search path

DT_RUNPATH

null-terminated library search path string

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

11.3.2.2 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 architecture specific part.

DT_ADDRRNGLO

Values from DT_ADDRRNGLO through DT_ADDRRNGHI are reserved for definition by an architecture specific part.

DT_AUXILIARY

Shared object to load before self

DT_FILTER

Shared object to get values from

DT_HIOS

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

DT_LOOS

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

DT_NUM

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

DT_POSFLAG_1

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

DT_RELCOUNT

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

DT_SYMINENT

Entry size of syminfo

DT_SYMINFO

Address of the Syminfo table.

DT_SYMINSZ

Size of syminfo table (in bytes)

DT_VALRNGHI

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

DT_VALRNGLO

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

DT_VERDEF

Address of version definition table

DT_VERDEFNUM

Number of version definitions

DT_VERNEED

Address of table with needed versions

DT_VERNEEDNUM

Number of needed versions

DT_VERSYM

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

12 C++ Class Representations

12.1 C++ Data Representation

Support for the C++ language shall be as specified in Itanium™ C++ ABI.

> **Note:** This document, although containing a few architecture specific matters, is written as a generic specification, to be usable by C++ implementations on a variety of architectures.

This section provides additional information to supplement Itanium™ C++ ABI. Many of the definitions in that document are made in terms of C++. This section provides addition explanations using C terms to avoid self-referential problems.

12.1.1 Class Representation

An object file generated by the compilation process for a C++ program shall contain several closely related internal objects, or Class Components, to represent each C++ Class. Such objects are not a visible part of the source code. Table 12-1 describes these Class Components at a high level.

Table 12-1 Class Components

Object	Contains
Class Data	All non-static Class members
Virtual Table	Information needed to dispatch virtual functions, access virtual base class subobjects and to access the RTTI information
RTTI	Run-Time Type Information used by the typeid and dynamic_cast operators, and exception handlers
Typeinfo Name	String representation of Class name
Construction Virtual Table	Information needed during construction and destruction of Classes with non-trivial inheritance relationships.
VTT	A table of virtual table pointers which holds the addresses of construction and non-construction virtual tables.

12.1.1.1 Virtual Table

Virtual tables are specified in Section 2.5.3 of Itanium™ C++ ABI.

Of the various categories of virtual table described in that specification, Category 1 (Leaf) is further described in Figure 12-1 and Category 2 (Non-virtual bases only) is further described in Figure 12-2. LSB conforming systems shall support these categories.

```
struct {
```



```
        ptrdiff_t       baseobject;
        const char      *typeinfo;
        fptr            virtfuncs[0];
};
```

Figure 12-1 Category 1 Virtual Table

```
struct {
        unsigned long   vcalloffset;
        ptrdiff_t       baseobject;
        const char      *typeinfo;
        fptr            virtfuncs[0];
};
```

Figure 12-2 Category 2 Virtual Table

This specification describes requirements for virtual tables of C++ classes using tables of the following form:

Table 12-2 Primary vtable for K (example)

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for K
vfunc[0]:	K::~K()
vfunc[1]:	K::~K()
vfunc[2]:	K::m1(int*)
vfunc[3]:	X::m2()
vfunc[4]:	`__cxa_pure_virtual()`
vfunc[5]:	NULL or X::m4(int)

Each row starting from 'vfunc[i]:' refers to a vtable entry 'vfunc[i]' of a class K, which is an entry for a virtual function A::m, where A is a base class of the class K as described in the Itanium™ C++ ABI. This specification requires implementations to interpret the vtable entry information in the following way:

1. A conforming implementation shall contain a vtable of the class K in the specified shared library;
2. The corresponding entry of this vtable 'vfunc[i]' shall be an entry for the virtual function A::m;
3. If the second column of the row contains `__cxa_pure_virtual()` the corresponding vtable entry of a LSB-conforming implementation shall contain `__cxa_pure_virtual()` or 'Y::m', where Y is the class K, the class A or a base class of the class K derived from the class A.

   Note: In this case virtual function A::m in class K is considered to be specified as pure virtual by this specification.

4. If the second column of the row contains 'X::m' the corresponding vtable entry of a LSB-conforming implementation shall contain 'Y::m', where Y is

the class K, the class X or a base class of the class K derived from the class X.

5. If the second column of the row contains 'NULL or X::m' the corresponding vtable entry of a LSB-conforming implementation shall contain NULL or 'Y::m', where Y is the class K, the class X or a base class of the class K derived from the class X.

Note: In this case virtual function A::m in class K is considered to be specified as inline by this specification.

An application may use any non-pure virtual function specified in this specification, and can expect the specified behavior irrespective of which particular method implements this functionality. An application may not use inline virtual functions at the binary level since its vtable entry may be NULL.

12.1.1.2 Run-Time Type Information

Each type used in a C++ program has a data structure associated with it that provide information about the type which is used at runtime. This Run Time Type Information (RTTI) is defined in section 2.9.5 in Itanium™ C++ ABI. Additional details about the layout of this data is provided here.

```
struct {
      void      *basevtable;
      char      *name;
};
```

Figure 12-3 Run-Time Type Information Prefix

```
struct {
      void      *basevtable;
      char      *name;
      void      *basetypeinfo[0];
};
```

Figure 12-4 Run-Time Type Information For Classes with no base class

```
struct {
      void      *basevtable;
      char      *name;
      void      *basetype;
      void      *basetypeinfo[0];
};
```

Figure 12-5 Run-Time Type Information for Classes with a single base class

```
struct base_type_info {
      char    *base_type;
      unsigned long   offset_flags;
};

struct {
      void    *basevtable;
      char    *name;
      unsigned int    flags;
      unsigned int    base_count;
      struct base_type_info base_info[0];
};
```

Figure 12-6 Run-Time Type Information for classes with multiple inheritance

```
struct {
```



```
        void    *basevtable;
        char    *name;
        unsigned int    flags;
        void    *pointee;
        void    *basetypeinfo[0];
};
```

Figure 12-7 Run-Time Type Information for pointer types

```
struct {
        void    *basevtable;
        char    *name;
        unsigned int    flags;
        void    *pointee;
        void    *context;
        void    *basetypeinfo[0];
};
```

Figure 12-8 Run-Time Type Information for pointer to member types

13 Symbol Mapping

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

13.1 Symbol Mapping

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

13.1.1 C++ Language

External symbol names in a C++ object file shall be encoded according to the "name mangling" rules described in the Itanium™ C++ ABI.

III Base Libraries

14 Base Libraries

14.1 Introduction

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

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

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

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

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

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

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

14.2 Program Interpreter

The Program Interpreter is specified in the appropriate architecture specific part of the LSB Core Specification.

14.3 Interfaces for libc

Table 14-1 defines the library name and shared object name for the libc library

Table 14-1 libc Definition

Library:	libc
SONAME:	See architecture specific part.

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

[LFS] Large File Support
[LSB] This Specification
[RPC + XDR] RFC 5531/4506 RPC & XDR
[SUSv2] SUSv2
[SUSv3] POSIX 1003.1-2001 (ISO/IEC 9945-2003)
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)
[SVID.4] SVID Issue 4

14.3.1 RPC

14.3.1.1 Interfaces for RPC

An LSB conforming implementation shall provide the generic functions for RPC specified in Table 14-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-2 libc - RPC Function Interfaces

authnone_create [SVID.4]	callrpc [RPC + XDR]	clnt_create [SVID.4]	clnt_pcreateerror [SVID.4]
clnt_perrno [SVID.4]	clnt_perror [SVID.4]	clnt_spcreateerror [SVID.4]	clnt_sperrno [SVID.4]
clnt_sperror [SVID.4]	clntraw_create [RPC + XDR]	clnttcp_create [RPC + XDR]	clntudp_bufcreate [RPC + XDR]
clntudp_create [RPC + XDR]	key_decryptsession [SVID.4]	pmap_getport [LSB]	pmap_set [LSB]
pmap_unset [LSB]	svc_getreqset [SVID.4]	svc_register [LSB]	svc_run [LSB]
svc_sendreply [LSB]	svcerr_auth [SVID.4]	svcerr_decode [SVID.4]	svcerr_noproc [SVID.4]
svcerr_noprog [SVID.4]	svcerr_progvers [SVID.4]	svcerr_systemerr [SVID.4]	svcerr_weakauth [SVID.4]
svcfd_create [RPC + XDR]	svcraw_create [RPC + XDR]	svctcp_create [LSB]	svcudp_create [LSB]
xdr_accepted_reply [SVID.4]	xdr_array [SVID.4]	xdr_bool [SVID.4]	xdr_bytes [SVID.4]
xdr_callhdr [SVID.4]	xdr_callmsg [SVID.4]	xdr_char [SVID.4]	xdr_double [SVID.4]

xdr_enum [SVID.4]	xdr_float [SVID.4]	xdr_free [SVID.4]	xdr_int [SVID.4]
xdr_long [SVID.4]	xdr_opaque [SVID.4]	xdr_opaque_auth [SVID.4]	xdr_pointer [SVID.4]
xdr_reference [SVID.4]	xdr_rejected_reply [SVID.4]	xdr_replymsg [SVID.4]	xdr_short [SVID.4]
xdr_string [SVID.4]	xdr_u_char [SVID.4]	xdr_u_int [LSB]	xdr_u_long [SVID.4]
xdr_u_short [SVID.4]	xdr_union [SVID.4]	xdr_vector [SVID.4]	xdr_void [SVID.4]
xdr_wrapstring [SVID.4]	xdrmem_create [SVID.4]	xdrrec_create [SVID.4]	xdrrec_endofrecord [RPC + XDR]
xdrrec_eof [SVID.4]	xdrrec_skiprecord [RPC + XDR]	xdrstdio_create [LSB]	

An LSB conforming implementation shall provide the generic deprecated functions for RPC specified in Table 14-3, 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 14-3 libc - RPC Deprecated Function Interfaces

key_decryptsession [SVID.4]			

14.3.2 Epoll

14.3.2.1 Interfaces for Epoll

An LSB conforming implementation shall provide the generic functions for Epoll specified in Table 14-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-4 libc - Epoll Function Interfaces

epoll_create(GLIBC_2.3.2) [LSB]	epoll_ctl(GLIBC_2.3.2) [LSB]	epoll_wait(GLIBC_2.3.2) [LSB]	

14.3.3 System Calls

14.3.3.1 Interfaces for System Calls

An LSB conforming implementation shall provide the generic functions for System Calls specified in Table 14-5, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-5 libc - System Calls Function Interfaces

__chk_fail(GLIBC_2.3.4) [LSB]	__fxstat [LSB]	__fxstatat(GLIBC_2.4) [LSB]	__getgroups_chk (GLIBC_2.4) [LSB]
__getpgid [LSB]	__lxstat [LSB]	__read_chk(GLIBC_2.4) [LSB]	__readlink_chk(GLIBC_2.4) [LSB]
__stack_chk_fail(GLIBC_2.4) [LSB]	__xmknod [LSB]	__xmknodat(GLIBC_2.4) [LSB]	__xstat [LSB]
access [SUSv4]	acct [LSB]	alarm [SUSv4]	backtrace [LSB]
backtrace_symbols [LSB]	backtrace_symbols_fd [LSB]	brk [SUSv2]	chdir [SUSv4]
chmod [SUSv4]	chown [SUSv4]	chroot [SUSv2]	clock [SUSv4]
close [SUSv4]	closedir [SUSv4]	creat [SUSv4]	dup [SUSv4]
dup2 [SUSv4]	execl [SUSv4]	execle [SUSv4]	execlp [SUSv4]
execv [SUSv4]	execve [SUSv4]	execvp [SUSv4]	exit [SUSv4]
faccessat(GLIBC_2.4) [SUSv4]	fchdir [SUSv4]	fchmod [SUSv4]	fchmodat(GLIBC_2.4) [SUSv4]
fchown [SUSv4]	fchownat(GLIBC_2.4) [SUSv4]	fcntl [LSB]	fdatasync [SUSv4]
fdopendir(GLIBC_2.4) [SUSv4]	fexecve [SUSv4]	flock [LSB]	fork [SUSv4]
fstatfs [LSB]	fstatvfs [SUSv4]	fsync [SUSv4]	ftime [SUSv3]
ftruncate [SUSv4]	futimens(GLIBC_2.6) [SUSv4]	futimes(GLIBC_2.3) [LSB]	getcontext [SUSv3]
getdtablesize [LSB]	getegid [SUSv4]	geteuid [SUSv4]	getgid [SUSv4]
getgroups [SUSv4]	getitimer [SUSv4]	getloadavg [LSB]	getpagesize [LSB]
getpgid [SUSv4]	getpgrp [SUSv4]	getpid [SUSv4]	getppid [SUSv4]
getpriority [SUSv4]	getrlimit [LSB]	getrusage [SUSv4]	getsid [SUSv4]
getuid [SUSv4]	getwd [SUSv3]	initgroups [LSB]	ioctl [LSB]
kill [LSB]	killpg [SUSv4]	lchown [SUSv4]	link [LSB]
linkat(GLIBC_2.4) [SUSv4]	lockf [SUSv4]	lseek [SUSv4]	lutimes(GLIBC_2.3) [LSB]
mkdir [SUSv4]	mkdirat(GLIBC_2.4) [SUSv4]	mkfifo [SUSv4]	mkfifoat(GLIBC_2.4) [SUSv4]

mlock [SUSv4]	mlockall [SUSv4]	mmap [SUSv4]	mprotect [SUSv4]
mremap [LSB]	msync [SUSv4]	munlock [SUSv4]	munlockall [SUSv4]
munmap [SUSv4]	nanosleep [SUSv4]	nice [SUSv4]	open [SUSv4]
openat(GLIBC_2.4) [SUSv4]	opendir [SUSv4]	pathconf [SUSv4]	pause [SUSv4]
pipe [SUSv4]	poll [SUSv4]	pread [SUSv4]	pselect [SUSv4]
ptrace [LSB]	pwrite [SUSv4]	read [SUSv4]	readdir [SUSv4]
readdir_r [SUSv4]	readlink [SUSv4]	readlinkat(GLIBC_2.4) [SUSv4]	readv [SUSv4]
rename [SUSv4]	renameat(GLIBC_2.4) [SUSv4]	rmdir [SUSv4]	sbrk [SUSv2]
sched_get_priority_max [SUSv4]	sched_get_priority_min [SUSv4]	sched_getaffinity (GLIBC_2.3.4) [LSB]	sched_getparam [SUSv4]
sched_getscheduler [SUSv4]	sched_rr_get_interval [SUSv4]	sched_setaffinity (GLIBC_2.3.4) [LSB]	sched_setparam [SUSv4]
sched_setscheduler [LSB]	sched_yield [SUSv4]	select [SUSv4]	setcontext [SUSv3]
setegid [SUSv4]	seteuid [SUSv4]	setgid [SUSv4]	setitimer [SUSv4]
setpgid [SUSv4]	setpgrp [SUSv4]	setpriority [SUSv4]	setregid [SUSv4]
setreuid [SUSv4]	setrlimit [LSB]	setrlimit64 [LFS]	setsid [SUSv4]
setuid [SUSv4]	sleep [SUSv4]	statfs [LSB]	statvfs [SUSv4]
stime [LSB]	symlink [SUSv4]	symlinkat(GLIBC_2.4) [SUSv4]	sync [SUSv4]
sysconf [LSB]	sysinfo [LSB]	time [SUSv4]	times [SUSv4]
truncate [SUSv4]	ulimit [SUSv4]	umask [SUSv4]	uname [SUSv4]
unlink [LSB]	unlinkat(GLIBC_2.4) [SUSv4]	utime [SUSv4]	utimensat(GLIBC_2.6) [SUSv4]
utimes [SUSv4]	vfork [SUSv3]	wait [SUSv4]	wait4 [LSB]
waitid [SUSv4]	waitpid [SUSv4]	write [SUSv4]	writev [SUSv4]

An LSB conforming implementation shall provide the generic deprecated functions for System Calls specified in Table 14-6, 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 14-6 libc - System Calls Deprecated Function Interfaces

fstatfs [LSB]	getdtablesize [LSB]	getpagesize [LSB]	getwd [SUSv3]
statfs [LSB]			

14.3.4 Standard I/O

14.3.4.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the generic functions for Standard I/O specified in Table 14-7, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-7 libc - Standard I/O Function Interfaces

_IO_feof [LSB]	_IO_getc [LSB]	_IO_putc [LSB]	_IO_puts [LSB]
__fgets_chk(GLIBC_2.4) [LSB]	__fgets_unlocked_chk(GLIBC_2.4) [LSB]	__fgetws_unlocked_chk(GLIBC_2.4) [LSB]	__fprintf_chk [LSB]
__printf_chk [LSB]	__snprintf_chk [LSB]	__sprintf_chk [LSB]	__vfprintf_chk [LSB]
__vprintf_chk [LSB]	__vsnprintf_chk [LSB]	__vsprintf_chk [LSB]	asprintf [LSB]
clearerr [SUSv4]	clearerr_unlocked [LSB]	ctermid [SUSv4]	dprintf [SUSv4]
fclose [SUSv4]	fdopen [SUSv4]	feof [SUSv4]	feof_unlocked [LSB]
ferror [SUSv4]	ferror_unlocked [LSB]	fflush [SUSv4]	fflush_unlocked [LSB]
fgetc [SUSv4]	fgetc_unlocked [LSB]	fgetpos [SUSv4]	fgets [SUSv4]
fgets_unlocked [LSB]	fgetwc_unlocked [LSB]	fgetws_unlocked [LSB]	fileno [SUSv4]
fileno_unlocked [LSB]	flockfile [SUSv4]	fopen [SUSv4]	fprintf [SUSv4]
fputc [SUSv4]	fputc_unlocked [LSB]	fputs [SUSv4]	fputs_unlocked [LSB]
fputwc_unlocked [LSB]	fputws_unlocked [LSB]	fread [SUSv4]	fread_unlocked [LSB]
freopen [SUSv4]	fscanf [LSB]	fseek [SUSv4]	fseeko [SUSv4]

fsetpos [SUSv4]	ftell [SUSv4]	ftello [SUSv4]	fwrite [SUSv4]
fwrite_unlocked [LSB]	getc [SUSv4]	getc_unlocked [SUSv4]	getchar [SUSv4]
getchar_unlocked [SUSv4]	getdelim [SUSv4]	getline [SUSv4]	getw [SUSv2]
getwc_unlocked [LSB]	getwchar_unlocked [LSB]	pclose [SUSv4]	popen [SUSv4]
printf [SUSv4]	putc [SUSv4]	putc_unlocked [SUSv4]	putchar [SUSv4]
putchar_unlocked [SUSv4]	puts [SUSv4]	putw [SUSv2]	putwc_unlocked [LSB]
putwchar_unlocked [LSB]	remove [SUSv4]	rewind [SUSv4]	rewinddir [SUSv4]
scanf [LSB]	seekdir [SUSv4]	setbuf [SUSv4]	setbuffer [LSB]
setvbuf [SUSv4]	snprintf [SUSv4]	sprintf [SUSv4]	sscanf [LSB]
telldir [SUSv4]	tempnam [SUSv4]	ungetc [SUSv4]	vasprintf [LSB]
vdprintf [SUSv4]	vfprintf [SUSv4]	vprintf [SUSv4]	vsnprintf [SUSv4]
vsprintf [SUSv4]			

An LSB conforming implementation shall provide the generic deprecated functions for Standard I/O specified in Table 14-8, 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 14-8 libc - Standard I/O Deprecated Function Interfaces

tempnam [SUSv4]			

An LSB conforming implementation shall provide the generic data interfaces for Standard I/O specified in Table 14-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-9 libc - Standard I/O Data Interfaces

stderr [SUSv4]	stdin [SUSv4]	stdout [SUSv4]	

14.3.5 Signal Handling

14.3.5.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the generic functions for Signal Handling specified in Table 14-10, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-10 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax [LSB]	__libc_current_sigrtmin [LSB]	__sigsetjmp [LSB]	__sysv_signal [LSB]
__xpg_sigpause [LSB]	bsd_signal [SUSv3]	psiginfo(GLIBC_2.10) [SUSv4]	psignal [SUSv4]
raise [SUSv4]	sigaction [SUSv4]	sigaddset [SUSv4]	sigaltstack [SUSv4]
sigandset [LSB]	sigdelset [SUSv4]	sigemptyset [SUSv4]	sigfillset [SUSv4]
sighold [SUSv4]	sigignore [SUSv4]	siginterrupt [SUSv4]	sigisemptyset [LSB]
sigismember [SUSv4]	siglongjmp [SUSv4]	signal [SUSv4]	sigorset [LSB]
sigpause [LSB]	sigpending [SUSv4]	sigprocmask [SUSv4]	sigqueue [SUSv4]
sigrelse [SUSv4]	sigreturn [LSB]	sigset [SUSv4]	sigsuspend [SUSv4]
sigtimedwait [SUSv4]	sigwait [SUSv4]	sigwaitinfo [SUSv4]	

An LSB conforming implementation shall provide the generic deprecated functions for Signal Handling specified in Table 14-11, 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 14-11 libc - Signal Handling Deprecated Function Interfaces

sigpause [LSB]			

An LSB conforming implementation shall provide the generic data interfaces for Signal Handling specified in Table 14-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-12 libc - Signal Handling Data Interfaces

_sys_siglist [LSB]			

14.3.6 Localization Functions

14.3.6.1 Interfaces for Localization Functions

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

Table 14-13 libc - Localization Functions Function Interfaces

bind_textdomain_codeset [LSB]	bindtextdomain [LSB]	catclose [SUSv4]	catgets [SUSv4]
catopen [SUSv4]	dcgettext [LSB]	dcngettext [LSB]	dgettext [LSB]
dngettext [LSB]	duplocale(GLIBC_2.3) [SUSv4]	freelocale(GLIBC_2.3) [SUSv4]	gettext [LSB]
iconv [SUSv4]	iconv_close [SUSv4]	iconv_open [SUSv4]	localeconv [SUSv4]
newlocale(GLIBC_2.3) [SUSv4]	ngettext [LSB]	nl_langinfo [SUSv4]	setlocale [SUSv4]
textdomain [LSB]	uselocale(GLIBC_2.3) [SUSv4]		

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

Table 14-14 libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr [LSB]			

14.3.7 Posix Spawn Option

14.3.7.1 Interfaces for Posix Spawn Option

An LSB conforming implementation shall provide the generic functions for Posix Spawn Option specified in Table 14-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-15 libc - Posix Spawn Option Function Interfaces

posix_spawn [SUSv4]	posix_spawn_file_actions_addclose [SUSv4]	posix_spawn_file_actions_adddup2 [SUSv4]	posix_spawn_file_actions_addopen [SUSv4]
posix_spawn_file_actions_destroy [SUSv4]	posix_spawn_file_actions_init [SUSv4]	posix_spawnattr_destroy [SUSv4]	posix_spawnattr_getflags [SUSv4]

posix_spawnattr_getpgroup [SUSv4]	posix_spawnattr_getschedparam [SUSv4]	posix_spawnattr_getschedpolicy [SUSv4]	posix_spawnattr_getsigdefault [SUSv4]
posix_spawnattr_getsigmask [SUSv4]	posix_spawnattr_init [SUSv4]	posix_spawnattr_setflags [SUSv4]	posix_spawnattr_setpgroup [SUSv4]
posix_spawnattr_setschedparam [SUSv4]	posix_spawnattr_setschedpolicy [SUSv4]	posix_spawnattr_setsigdefault [SUSv4]	posix_spawnattr_setsigmask [SUSv4]
posix_spawnp [SUSv4]			

14.3.8 Posix Advisory Option

14.3.8.1 Interfaces for Posix Advisory Option

An LSB conforming implementation shall provide the generic functions for Posix Advisory Option specified in Table 14-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-16 libc - Posix Advisory Option Function Interfaces

posix_fadvise [SUSv4]	posix_fallocate [SUSv4]	posix_madvise [SUSv4]	posix_memalign [SUSv4]

14.3.9 Socket Interface

14.3.9.1 Interfaces for Socket Interface

An LSB conforming implementation shall provide the generic functions for Socket Interface specified in Table 14-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-17 libc - Socket Interface Function Interfaces

__gethostname_chk(GLIBC_2.4) [LSB]	__h_errno_location [LSB]	__recv_chk(GLIBC_2.4) [LSB]	__recvfrom_chk(GLIBC_2.4) [LSB]
accept [SUSv4]	bind [SUSv4]	bindresvport [LSB]	connect [SUSv4]
freeifaddrs(GLIBC_2.3) [LSB]	gethostid [SUSv4]	gethostname [SUSv4]	getifaddrs(GLIBC_2.3) [LSB]
getpeername [SUSv4]	getsockname [SUSv4]	getsockopt [LSB]	if_freenameindex [SUSv4]
if_indextoname [SUSv4]	if_nameindex [SUSv4]	if_nametoindex [SUSv4]	listen [SUSv4]
recv [SUSv4]	recvfrom [SUSv4]	recvmsg [SUSv4]	send [SUSv4]

sendmsg [SUSv4]	sendto [SUSv4]	setsockopt [LSB]	shutdown [SUSv4]
sockatmark [SUSv4]	socket [SUSv4]	socketpair [SUSv4]	

An LSB conforming implementation shall provide the generic data interfaces for Socket Interface specified in Table 14-18, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-18 libc - Socket Interface Data Interfaces

in6addr_any [SUSv3]	in6addr_loopback [SUSv3]		

14.3.10 Wide Characters

14.3.10.1 Interfaces for Wide Characters

An LSB conforming implementation shall provide the generic functions for Wide Characters specified in Table 14-19, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-19 libc - Wide Characters Function Interfaces

__fgetws_chk(GLIBC_2.4) [LSB]	__fwprintf_chk(GLIBC_2.4) [LSB]	__mbsnrtowcs_chk(GLIBC_2.4) [LSB]	__mbsrtowcs_chk(GLIBC_2.4) [LSB]
__mbstowcs_chk(GLIBC_2.4) [LSB]	__swprintf_chk(GLIBC_2.4) [LSB]	__vfwprintf_chk(GLIBC_2.4) [LSB]	__vswprintf_chk(GLIBC_2.4) [LSB]
__vwprintf_chk(GLIBC_2.4) [LSB]	__wcpcpy_chk(GLIBC_2.4) [LSB]	__wcpncpy_chk(GLIBC_2.4) [LSB]	__wcrtomb_chk(GLIBC_2.4) [LSB]
__wcscat_chk(GLIBC_2.4) [LSB]	__wcscpy_chk(GLIBC_2.4) [LSB]	__wcsncat_chk(GLIBC_2.4) [LSB]	__wcsncpy_chk(GLIBC_2.4) [LSB]
__wcsnrtombs_chk(GLIBC_2.4) [LSB]	__wcsrtombs_chk(GLIBC_2.4) [LSB]	__wcstod_internal [LSB]	__wcstof_internal [LSB]
__wcstol_internal [LSB]	__wcstold_internal [LSB]	__wcstombs_chk(GLIBC_2.4) [LSB]	__wcstoul_internal [LSB]
__wctomb_chk(GLIBC_2.4) [LSB]	__wmemcpy_chk(GLIBC_2.4) [LSB]	__wmemmove_chk(GLIBC_2.4) [LSB]	__wmempcpy_chk(GLIBC_2.4) [LSB]
__wmemset_chk(GLIBC_2.4) [LSB]	__wprintf_chk(GLIBC_2.4) [LSB]	btowc [SUSv4]	fgetwc [SUSv4]
fgetws [SUSv4]	fputwc [SUSv4]	fputws [SUSv4]	fwide [SUSv4]

fwprintf [SUSv4]	fwscanf [LSB]	getwc [SUSv4]	getwchar [SUSv4]
iswalnum_l(GLIBC_2.3) [SUSv4]	iswalpha_l(GLIBC_2.3) [SUSv4]	iswblank_l(GLIBC_2.3) [SUSv4]	iswcntrl_l(GLIBC_2.3) [SUSv4]
iswctype_l(GLIBC_2.3) [SUSv4]	iswdigit_l(GLIBC_2.3) [SUSv4]	iswgraph_l(GLIBC_2.3) [SUSv4]	iswlower_l(GLIBC_2.3) [SUSv4]
iswprint_l(GLIBC_2.3) [SUSv4]	iswpunct_l(GLIBC_2.3) [SUSv4]	iswspace_l(GLIBC_2.3) [SUSv4]	iswupper_l(GLIBC_2.3) [SUSv4]
iswxdigit_l(GLIBC_2.3) [SUSv4]	mblen [SUSv4]	mbrlen [SUSv4]	mbrtowc [SUSv4]
mbsinit [SUSv4]	mbsnrtowcs [SUSv4]	mbsrtowcs [SUSv4]	mbstowcs [SUSv4]
mbtowc [SUSv4]	putwc [SUSv4]	putwchar [SUSv4]	swprintf [SUSv4]
swscanf [LSB]	towctrans [SUSv4]	towctrans_l(GLIBC_2.3) [SUSv4]	towlower [SUSv4]
towlower_l(GLIBC_2.3) [SUSv4]	towupper [SUSv4]	towupper_l(GLIBC_2.3) [SUSv4]	ungetwc [SUSv4]
vfwprintf [SUSv4]	vfwscanf [LSB]	vswprintf [SUSv4]	vswscanf [LSB]
vwprintf [SUSv4]	vwscanf [LSB]	wcpcpy [SUSv4]	wcpncpy [SUSv4]
wcrtomb [SUSv4]	wcscasecmp [SUSv4]	wcscasecmp_l(GLIBC_2.3) [SUSv4]	wcscat [SUSv4]
wcschr [SUSv4]	wcscmp [SUSv4]	wcscoll [SUSv4]	wcscoll_l(GLIBC_2.3) [SUSv4]
wcscpy [SUSv4]	wcscspn [SUSv4]	wcsdup [SUSv4]	wcsftime [SUSv4]
wcslen [SUSv4]	wcsncasecmp [SUSv4]	wcsncasecmp_l(GLIBC_2.3) [SUSv4]	wcsncat [SUSv4]
wcsncmp [SUSv4]	wcsncpy [SUSv4]	wcsnlen [SUSv4]	wcsnrtombs [SUSv4]
wcspbrk [SUSv4]	wcsrchr [SUSv4]	wcsrtombs [SUSv4]	wcsspn [SUSv4]
wcsstr [SUSv4]	wcstod [SUSv4]	wcstof [SUSv4]	wcstoimax [SUSv4]
wcstok [SUSv4]	wcstol [SUSv4]	wcstold [SUSv4]	wcstoll [SUSv4]

wcstombs [SUSv4]	wcstoq [LSB]	wcstoul [SUSv4]	wcstoull [SUSv4]
wcstoumax [SUSv4]	wcstouq [LSB]	wcswcs [SUSv3]	wcswidth [SUSv4]
wcsxfrm [SUSv4]	wcsxfrm_l(GLIBC_2.3) [SUSv4]	wctob [SUSv4]	wctomb [SUSv4]
wctrans [SUSv4]	wctrans_l(GLIBC_2.3) [SUSv4]	wctype [SUSv4]	wctype_l(GLIBC_2.3) [SUSv4]
wcwidth [SUSv4]	wmemchr [SUSv4]	wmemcmp [SUSv4]	wmemcpy [SUSv4]
wmemmove [SUSv4]	wmemset [SUSv4]	wprintf [SUSv4]	wscanf [LSB]

14.3.11 String Functions

14.3.11.1 Interfaces for String Functions

An LSB conforming implementation shall provide the generic functions for String Functions specified in Table 14-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-20 libc - String Functions Function Interfaces

__memcpy_chk(GLIBC_2.3.4) [LSB]	__memmove_chk(GLIBC_2.3.4) [LSB]	__mempcpy [LSB]	__mempcpy_chk(GLIBC_2.3.4) [LSB]
__memset_chk(GLIBC_2.3.4) [LSB]	__rawmemchr [LSB]	__stpcpy [LSB]	__stpcpy_chk(GLIBC_2.3.4) [LSB]
__stpncpy_chk(GLIBC_2.4) [LSB]	__strcat_chk(GLIBC_2.3.4) [LSB]	__strcpy_chk(GLIBC_2.3.4) [LSB]	__strdup [LSB]
__strncat_chk(GLIBC_2.3.4) [LSB]	__strncpy_chk(GLIBC_2.3.4) [LSB]	__strtod_internal [LSB]	__strtof_internal [LSB]
__strtok_r [LSB]	__strtol_internal [LSB]	__strtold_internal [LSB]	__strtoll_internal [LSB]
__strtoul_internal [LSB]	__strtoull_internal [LSB]	__xpg_strerror_r(GLIBC_2.3.4) [LSB]	bcmp [SUSv3]
bcopy [SUSv3]	bzero [SUSv3]	ffs [SUSv4]	index [SUSv3]
memccpy [SUSv4]	memchr [SUSv4]	memcmp [SUSv4]	memcpy [SUSv4]
memmove [SUSv4]	memrchr [LSB]	memset [SUSv4]	rindex [SUSv3]

stpcpy [SUSv4]	stpncpy [SUSv4]	strcasecmp [SUSv4]	strcasecmp_l(GLIBC_2.3) [SUSv4]
strcasestr [LSB]	strcat [SUSv4]	strchr [SUSv4]	strcmp [SUSv4]
strcoll [SUSv4]	strcoll_l(GLIBC_2.3) [SUSv4]	strcpy [SUSv4]	strcspn [SUSv4]
strdup [SUSv4]	strerror [SUSv4]	strerror_l(GLIBC_2.6) [SUSv4]	strerror_r [LSB]
strfmon [SUSv4]	strfmon_l(GLIBC_2.3) [SUSv4]	strftime [SUSv4]	strftime_l(GLIBC_2.3) [SUSv4]
strlen [SUSv4]	strncasecmp [SUSv4]	strncasecmp_l(GLIBC_2.3) [SUSv4]	strncat [SUSv4]
strncmp [SUSv4]	strncpy [SUSv4]	strndup [SUSv4]	strnlen [SUSv4]
strpbrk [SUSv4]	strptime [LSB]	strrchr [SUSv4]	strsep [LSB]
strsignal [SUSv4]	strspn [SUSv4]	strstr [SUSv4]	strtof [SUSv4]
strtoimax [SUSv4]	strtok [SUSv4]	strtok_r [SUSv4]	strtold [SUSv4]
strtoll [SUSv4]	strtoq [LSB]	strtoull [SUSv4]	strtoumax [SUSv4]
strtouq [LSB]	strxfrm [SUSv4]	strxfrm_l(GLIBC_2.3) [SUSv4]	swab [SUSv4]

An LSB conforming implementation shall provide the generic deprecated functions for String Functions specified in Table 14-21, 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 14-21 libc - String Functions Deprecated Function Interfaces

strerror_r [LSB]			

14.3.12 IPC Functions

14.3.12.1 Interfaces for IPC Functions

An LSB conforming implementation shall provide the generic functions for IPC Functions specified in Table 14-22, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-22 libc - IPC Functions Function Interfaces

ftok [SUSv4]	msgctl [SUSv4]	msgget [SUSv4]	msgrcv [SUSv4]
msgsnd [SUSv4]	semctl [SUSv4]	semget [SUSv4]	semop [SUSv4]

shmat [SUSv4]	shmctl [SUSv4]	shmdt [SUSv4]	shmget [SUSv4]

14.3.13 Regular Expressions

14.3.13.1 Interfaces for Regular Expressions

An LSB conforming implementation shall provide the generic functions for Regular Expressions specified in Table 14-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-23 libc - Regular Expressions Function Interfaces

regcomp [SUSv4]	regerror [SUSv4]	regexec [LSB]	regfree [SUSv4]

14.3.14 Character Type Functions

14.3.14.1 Interfaces for Character Type Functions

An LSB conforming implementation shall provide the generic functions for Character Type Functions specified in Table 14-24, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-24 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 [SUSv4]	_toupper [SUSv4]	isalnum [SUSv4]	isalnum_l(GLIBC_2.3) [SUSv4]
isalpha [SUSv4]	isalpha_l(GLIBC_2.3) [SUSv4]	isascii [SUSv4]	isblank_l(GLIBC_2.3) [SUSv4]
iscntrl [SUSv4]	iscntrl_l(GLIBC_2.3) [SUSv4]	isdigit [SUSv4]	isdigit_l(GLIBC_2.3) [SUSv4]
isgraph [SUSv4]	isgraph_l(GLIBC_2.3) [SUSv4]	islower [SUSv4]	islower_l(GLIBC_2.3) [SUSv4]
isprint [SUSv4]	isprint_l(GLIBC_2.3) [SUSv4]	ispunct [SUSv4]	ispunct_l(GLIBC_2.3) [SUSv4]
isspace [SUSv4]	isspace_l(GLIBC_2.3) [SUSv4]	isupper [SUSv4]	isupper_l(GLIBC_2.3) [SUSv4]
iswalnum [SUSv4]	iswalpha [SUSv4]	iswblank [SUSv4]	iswcntrl [SUSv4]
iswctype [SUSv4]	iswdigit [SUSv4]	iswgraph [SUSv4]	iswlower [SUSv4]
iswprint [SUSv4]	iswpunct [SUSv4]	iswspace [SUSv4]	iswupper [SUSv4]
iswxdigit [SUSv4]	isxdigit [SUSv4]	isxdigit_l(GLIBC_2.3) [SUSv4]	toascii [SUSv4]

tolower [SUSv4]	tolower_l(GLIBC_2.3) [SUSv4]	toupper [SUSv4]	toupper_l(GLIBC_2.3) [SUSv4]

14.3.15 Time Manipulation

14.3.15.1 Interfaces for Time Manipulation

An LSB conforming implementation shall provide the generic functions for Time Manipulation specified in Table 14-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-25 libc - Time Manipulation Function Interfaces

adjtime [LSB]	asctime [SUSv4]	asctime_r [SUSv4]	ctime [SUSv4]
ctime_r [SUSv4]	difftime [SUSv4]	gmtime [SUSv4]	gmtime_r [SUSv4]
localtime [SUSv4]	localtime_r [SUSv4]	mktime [SUSv4]	tzset [SUSv4]
ualarm [SUSv3]			

An LSB conforming implementation shall provide the generic data interfaces for Time Manipulation specified in Table 14-26, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-26 libc - Time Manipulation Data Interfaces

__daylight [LSB]	__timezone [LSB]	__tzname [LSB]	daylight [SUSv4]
timezone [SUSv4]	tzname [SUSv4]		

14.3.16 Terminal Interface Functions

14.3.16.1 Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the generic functions for Terminal Interface Functions specified in Table 14-27, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-27 libc - Terminal Interface Functions Function Interfaces

cfgetispeed [SUSv4]	cfgetospeed [SUSv4]	cfmakeraw [LSB]	cfsetispeed [SUSv4]
cfsetospeed [SUSv4]	cfsetspeed [LSB]	tcdrain [SUSv4]	tcflow [SUSv4]
tcflush [SUSv4]	tcgetattr [SUSv4]	tcgetpgrp [SUSv4]	tcgetsid [SUSv4]
tcsendbreak [SUSv4]	tcsetattr [SUSv4]	tcsetpgrp [SUSv4]	

14.3.17 System Database Interface

14.3.17.1 Interfaces for System Database Interface

An LSB conforming implementation shall provide the generic functions for System Database Interface specified in Table 14-28, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-28 libc - System Database Interface Function Interfaces

endgrent [SUSv4]	endprotoent [SUSv4]	endpwent [SUSv4]	endservent [SUSv4]
endutent [LSB]	endutxent [SUSv4]	getgrent [SUSv4]	getgrent_r [LSB]
getgrgid [SUSv4]	getgrgid_r [SUSv4]	getgrnam [SUSv4]	getgrnam_r [SUSv4]
getgrouplist [LSB]	gethostbyaddr [SUSv3]	gethostbyaddr_r [LSB]	gethostbyname [SUSv3]
gethostbyname2 [LSB]	gethostbyname2_r [LSB]	gethostbyname_r [LSB]	getprotobyname [SUSv4]
getprotobyname_r [LSB]	getprotobynumber [SUSv4]	getprotobynumber_r [LSB]	getprotoent [SUSv4]
getprotoent_r [LSB]	getpwent [SUSv4]	getpwent_r [LSB]	getpwnam [SUSv4]
getpwnam_r [SUSv4]	getpwuid [SUSv4]	getpwuid_r [SUSv4]	getservbyname [SUSv4]
getservbyname_r [LSB]	getservbyport [SUSv4]	getservbyport_r [LSB]	getservent [SUSv4]
getservent_r [LSB]	getutent [LSB]	getutent_r [LSB]	getutxent [SUSv4]
getutxid [SUSv4]	getutxline [SUSv4]	pututxline [SUSv4]	setgrent [SUSv4]
setgroups [LSB]	setprotoent [SUSv4]	setpwent [SUSv4]	setservent [SUSv4]
setutent [LSB]	setutxent [SUSv4]	utmpname [LSB]	

An LSB conforming implementation shall provide the generic deprecated functions for System Database Interface specified in Table 14-29, 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 14-29 libc - System Database Interface Deprecated Function Interfaces

gethostbyaddr [SUSv3]	gethostbyaddr_r [LSB]	gethostbyname [SUSv3]	gethostbyname2 [LSB]
gethostbyname2_r [LSB]	gethostbyname_r [LSB]		

14.3.18 Language Support

14.3.18.1 Interfaces for Language Support

An LSB conforming implementation shall provide the generic functions for Language Support specified in Table 14-30, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-30 libc - Language Support Function Interfaces

__libc_start_main [LSB]	__register_atfork (GLIBC_2.3.2) [LSB]		

14.3.19 Large File Support

14.3.19.1 Interfaces for Large File Support

An LSB conforming implementation shall provide the generic functions for Large File Support specified in Table 14-31, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-31 libc - Large File Support Function Interfaces

__fxstat64 [LSB]	__fxstatat64(GLIBC_2.4) [LSB]	__lxstat64 [LSB]	__xstat64 [LSB]
creat64 [LFS]	fgetpos64 [LFS]	fopen64 [LFS]	freopen64 [LFS]
fseeko64 [LFS]	fsetpos64 [LFS]	fstatfs64 [LSB]	fstatvfs64 [LFS]
ftello64 [LFS]	ftruncate64 [LFS]	ftw64 [LFS]	getrlimit64 [LFS]
lockf64 [LFS]	lseek64 [LFS]	mkstemp64 [LSB]	mmap64 [LFS]
nftw64 [LFS]	open64 [LFS]	openat64(GLIBC_2.4) [LSB]	posix_fadvise64 [LSB]
posix_fallocate64 [LSB]	pread64 [LSB]	pwrite64 [LSB]	readdir64 [LFS]
readdir64_r [LSB]	statfs64 [LSB]	statvfs64 [LFS]	tmpfile64 [LFS]
truncate64 [LFS]			

An LSB conforming implementation shall provide the generic deprecated functions for Large File Support specified in Table 14-32, 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 14-32 libc - Large File Support Deprecated Function Interfaces

fstatfs64 [LSB]	statfs64 [LSB]		

14.3.20 Inotify

14.3.20.1 Interfaces for Inotify

An LSB conforming implementation shall provide the generic functions for Inotify specified in Table 14-33, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-33 libc - Inotify Function Interfaces

inotify_add_watch(GLIBC_2.4) [LSB]	inotify_init(GLIBC_2.4) [LSB]	inotify_rm_watch(GLIBC_2.4) [LSB]	

14.3.21 Standard Library

14.3.21.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the generic functions for Standard Library specified in Table 14-34, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-34 libc - Standard Library Function Interfaces

_Exit [SUSv4]	__assert_fail [LSB]	__confstr_chk(GLIBC_2.4) [LSB]	__cxa_atexit [LSB]
__cxa_finalize [LSB]	__errno_location [LSB]	__fpending [LSB]	__getcwd_chk(GLIBC_2.4) [LSB]
__getlogin_r_chk(GLIBC_2.4) [LSB]	__getpagesize [LSB]	__isinf [LSB]	__isinff [LSB]
__isinfl [LSB]	__isnan [LSB]	__isnanf [LSB]	__isnanl [LSB]
__pread64_chk(GLIBC_2.4) [LSB]	__pread_chk(GLIBC_2.4) [LSB]	__realpath_chk(GLIBC_2.4) [LSB]	__sysconf [LSB]
__syslog_chk(GLIBC_2.4) [LSB]	__ttyname_r_chk(GLIBC_2.4) [LSB]	__vsyslog_chk(GLIBC_2.4) [LSB]	__xpg_basename [LSB]
_exit [SUSv4]	_longjmp [SUSv4]	_setjmp [SUSv4]	a64l [SUSv4]

abort [SUSv4]	abs [SUSv4]	alphasort [SUSv4]	alphasort64 [LSB]
argz_add [LSB]	argz_add_sep [LSB]	argz_append [LSB]	argz_count [LSB]
argz_create [LSB]	argz_create_sep [LSB]	argz_delete [LSB]	argz_extract [LSB]
argz_insert [LSB]	argz_next [LSB]	argz_replace [LSB]	argz_stringify [LSB]
atof [SUSv4]	atoi [SUSv4]	atol [SUSv4]	atoll [SUSv4]
basename [LSB]	bsearch [SUSv4]	calloc [SUSv4]	closelog [SUSv4]
confstr [SUSv4]	cuserid [SUSv2]	daemon [LSB]	dirfd [SUSv4]
dirname [SUSv4]	div [SUSv4]	dl_iterate_phdr [LSB]	drand48 [SUSv4]
drand48_r [LSB]	ecvt [SUSv3]	envz_add [LSB]	envz_entry [LSB]
envz_get [LSB]	envz_merge [LSB]	envz_remove [LSB]	envz_strip [LSB]
erand48 [SUSv4]	erand48_r [LSB]	err [LSB]	error [LSB]
errx [LSB]	fcvt [SUSv3]	fmemopen [SUSv4]	fmtmsg [SUSv4]
fnmatch [LSB]	fpathconf [SUSv4]	free [SUSv4]	freeaddrinfo [SUSv4]
ftrylockfile [SUSv4]	ftw [SUSv4]	funlockfile [SUSv4]	gai_strerror [SUSv4]
gcvt [SUSv3]	getaddrinfo [SUSv4]	getcwd [LSB]	getdate [SUSv4]
getdomainname [LSB]	getenv [SUSv4]	getlogin [SUSv4]	getlogin_r [SUSv4]
getnameinfo [SUSv4]	getopt [LSB]	getopt_long [LSB]	getopt_long_only [LSB]
getsubopt [SUSv4]	gettimeofday [SUSv4]	glob [SUSv4]	glob64 [LSB]
globfree [SUSv4]	globfree64 [LSB]	grantpt [SUSv4]	hcreate [SUSv4]
hcreate_r [LSB]	hdestroy [SUSv4]	hdestroy_r [LSB]	hsearch [SUSv4]
hsearch_r [LSB]	htonl [SUSv4]	htons [SUSv4]	imaxabs [SUSv4]
imaxdiv [SUSv4]	inet_addr [SUSv4]	inet_aton [LSB]	inet_ntoa [SUSv4]
inet_ntop [SUSv4]	inet_pton [SUSv4]	initstate [SUSv4]	initstate_r [LSB]

insque [SUSv4]	isatty [SUSv4]	isblank [SUSv4]	jrand48 [SUSv4]
jrand48_r [LSB]	l64a [SUSv4]	labs [SUSv4]	lcong48 [SUSv4]
lcong48_r [LSB]	ldiv [SUSv4]	lfind [SUSv4]	llabs [SUSv4]
lldiv [SUSv4]	longjmp [SUSv4]	lrand48 [SUSv4]	lrand48_r [LSB]
lsearch [SUSv4]	makecontext [SUSv3]	malloc [SUSv4]	memmem [LSB]
mkdtemp [SUSv4]	mkstemp [SUSv4]	mktemp [SUSv3]	mrand48 [SUSv4]
mrand48_r [LSB]	nftw [SUSv4]	nrand48 [SUSv4]	nrand48_r [LSB]
ntohl [SUSv4]	ntohs [SUSv4]	open_memstrea m [SUSv4]	open_wmemstre am(GLIBC_2.4) [SUSv4]
openlog [SUSv4]	perror [SUSv4]	posix_openpt [SUSv4]	ptsname [SUSv4]
putenv [SUSv4]	qsort [SUSv4]	rand [SUSv4]	rand_r [SUSv4]
random [SUSv4]	random_r [LSB]	realloc [SUSv4]	realpath [SUSv4]
remque [SUSv4]	scandir [SUSv4]	scandir64 [LSB]	seed48 [SUSv4]
seed48_r [LSB]	sendfile [LSB]	sendfile64(GLIB C_2.3) [LSB]	setenv [SUSv4]
sethostname [LSB]	setlogmask [SUSv4]	setstate [SUSv4]	setstate_r [LSB]
srand [SUSv4]	srand48 [SUSv4]	srand48_r [LSB]	srandom [SUSv4]
srandom_r [LSB]	strtod [SUSv4]	strtol [SUSv4]	strtoul [SUSv4]
swapcontext [SUSv3]	syslog [SUSv4]	system [LSB]	tdelete [SUSv4]
tfind [SUSv4]	tmpfile [SUSv4]	tmpnam [SUSv4]	tsearch [SUSv4]
ttyname [SUSv4]	ttyname_r [SUSv4]	twalk [SUSv4]	unlockpt [SUSv4]
unsetenv [SUSv4]	usleep [SUSv3]	verrx [LSB]	vfscanf [LSB]
vscanf [LSB]	vsscanf [LSB]	vsyslog [LSB]	warn [LSB]
warnx [LSB]	wordexp [SUSv4]	wordfree [SUSv4]	

An LSB conforming implementation shall provide the generic deprecated functions for Standard Library specified in Table 14-35, with the full mandatory functionality as described in the referenced underlying specification.

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

Table 14-35 libc - Standard Library Deprecated Function Interfaces

basename [LSB]	getdomainname [LSB]	inet_aton [LSB]	tmpnam [SUSv4]

An LSB conforming implementation shall provide the generic data interfaces for Standard Library specified in Table 14-36, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-36 libc - Standard Library Data Interfaces

__environ [LSB]	_environ [LSB]	_sys_errlist [LSB]	environ [SUSv4]
getdate_err [SUSv4]	optarg [SUSv4]	opterr [SUSv4]	optind [SUSv4]
optopt [SUSv4]			

14.3.22 GNU Extensions for libc

14.3.22.1 Interfaces for GNU Extensions for libc

An LSB conforming implementation shall provide the generic functions for GNU Extensions for libc specified in Table 14-37, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-37 libc - GNU Extensions for libc Function Interfaces

gnu_get_libc_release [LSB]	gnu_get_libc_version [LSB]		

14.4 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.

14.4.1 argz.h

```
typedef int error_t;
extern error_t argz_add(char **argz, size_t * argz_len, const char
*str);
extern error_t argz_add_sep(char **argz, size_t * argz_len,
                            const char *str, int sep);
extern error_t argz_append(char **argz, size_t * argz_len, const
char *buf,
                           size_t buf_len);
extern size_t argz_count(const char *argz, size_t * argz_len);
extern error_t argz_create(char *const argv[], char **argz,
                           size_t * argz_len);
extern error_t argz_create_sep(const char *str, int sep, char
**argz,
                               size_t * argz_len);
extern void argz_delete(char **argz, size_t * argz_len, char
*entry);
extern void argz_extract(const char *argz, size_t argz_len, char
**argv);
extern error_t argz_insert(char **argz_insert, size_t * argz_len,
                           char *before, const char *entry);
extern char argz_next(const char *argz, size_t argz_len,
                      const char *entry);
extern error_t argz_replace(char **argz, size_t * argz_len,
                            const char *str, const char *with,
                            unsigned int *replace_count);
extern void argz_stringify(char *argz, size_t argz_len, int sep);
```

14.4.2 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 *__buf,
                             socklen_t __len);
extern int inet_pton(int __af, const char *__cp, void *__buf);
extern uint32_t ntohl(uint32_t);
extern uint16_t ntohs(uint16_t);
```

14.4.3 assert.h

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

extern void __assert_fail(const char *__assertion, const char
*__file,
                          unsigned int __line, const char *__function);
```

14.4.4 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 001000
#define C_ISGID 002000
#define C_ISUID 004000
#define C_ISFIFO       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"
```

14.4.5 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 isalnum_l(int c, locale_t locale);
extern int isalpha(int);
extern int isalpha_l(int c, locale_t locale);
extern int isascii(int __c);
extern int isblank(int);
extern int isblank_l(int c, locale_t locale);
extern int iscntrl(int);
extern int iscntrl_l(int c, locale_t locale);
extern int isdigit(int);
extern int isdigit_l(int c, locale_t locale);
extern int isgraph(int);
extern int isgraph_l(int c, locale_t locale);
extern int islower(int);
extern int islower_l(int c, locale_t locale);
extern int isprint(int);
extern int isprint_l(int c, locale_t locale);
extern int ispunct(int);
extern int ispunct_l(int c, locale_t locale);
extern int isspace(int);
extern int isspace_l(int c, locale_t locale);
extern int isupper(int);
extern int isupper_l(int c, locale_t locale);
extern int isxdigit(int);
extern int isxdigit_l(int c, locale_t locale);
extern int toascii(int __c);
extern int tolower(int __c);
extern int tolower_l(int c, locale_t locale);
extern int toupper(int __c);
extern int toupper_l(int c, locale_t locale);
```

14.4.6 dirent.h

```
#define MAXNAMLEN      NAME_MAX

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);
```

14.4.7 elf.h

```
#define ELFMAG1 'E'
#define ELFMAG3 'F'
#define ELFMAG2 'L'
#define ELF64_R_INFO(sym, type) ((((Elf64_Xword) (sym)) << 32) +
(type))
#define ELF32_ST_INFO(bind,type)        (((bind) << 4) + ((type) &
0xf))
#define ELF32_R_INFO(sym, type) (((sym) << 8) + ((type) & 0xff))
#define ELF32_M_INFO(sym, size) (((sym) << 8) + (unsigned char)
(size))
#define ELF32_ST_BIND(val)      (((unsigned char) (val)) >> 4)
#define ELF64_R_TYPE(i) ((i) & 0xffffffff)
#define ELF64_R_SYM(i)  ((i) >> 32)
#define ELF32_M_SYM(info)       ((info) >> 8)
#define ELF32_ST_VISIBILITY(o)  ((o) & 0x03)
#define ELF32_M_SIZE(info)      ((unsigned char) (info))
#define ELF32_ST_TYPE(val)      ((val) & 0xf)
#define ELF32_R_TYPE(val)       ((val) & 0xff)
#define ELF32_R_SYM(val)        ((val) >> 8)
#define PF_X    (1 << 0)
#define SHF_WRITE       (1 << 0)
#define PF_W    (1 << 1)
#define SHF_ALLOC       (1 << 1)
#define SHF_TLS (1 << 10)
#define PF_R    (1 << 2)
#define SHF_EXECINSTR   (1 << 2)
```



```
#define SHF_MERGE       (1 << 4)
#define SHF_STRINGS     (1 << 5)
#define SHF_INFO_LINK   (1 << 6)
#define SHF_LINK_ORDER  (1 << 7)
#define SHF_OS_NONCONFORMING    (1 << 8)
#define SHF_GROUP       (1 << 9)
#define EI_NIDENT       (16)
#define DT_ADDRTAGIDX(tag)      (DT_ADDRRNGHI - (tag))
#define DT_IA_64_PLT_RESERVE    (DT_LOPROC + 0)
#define DT_PPC64_GLINK  (DT_LOPROC + 0)
#define DT_PPC_GOT      (DT_LOPROC + 0)
#define DT_PPC64_OPD    (DT_LOPROC + 1)
#define DT_PPC64_OPDSZ  (DT_LOPROC + 2)
#define DT_VALTAGIDX(tag)       (DT_VALRNGHI - (tag))
#define DT_VERSIONTAGIDX(tag)   (DT_VERNEEDNUM - (tag))
#define PT_IA_64_ARCHEXT        (PT_LOPROC + 0)
#define PT_IA_64_UNWIND (PT_LOPROC + 1)
#define SHT_IA_64_EXT   (SHT_LOPROC + 0)
#define SHT_IA_64_UNWIND        (SHT_LOPROC + 1)
#define DT_NULL 0
#define EI_MAG0 0
#define ELFCLASSNONE    0
#define ELFDATANONE     0
#define ELFOSABI_NONE   0
#define ELFOSABI_SYSV   0
#define ELF_NOTE_OS_LINUX       0
#define EM_NONE 0
#define ET_NONE 0
#define EV_NONE 0
#define PT_NULL 0
#define R_386_NONE      0
#define R_390_NONE      0
#define R_PPC_NONE      0
#define R_X86_64_NONE   0
#define SHN_UNDEF       0
#define SHT_NULL        0
#define STB_LOCAL       0
#define STN_UNDEF       0
#define STT_NOTYPE      0
#define STV_DEFAULT     0
#define SYMINFO_NONE    0
#define R_IA64_NONE     0x00
#define DF_1_NOW        0x00000001
#define DF_ORIGIN       0x00000001      /* Object may use DF_ORIGIN
*/
#define DF_P1_LAZYLOAD  0x00000001
#define DTF_1_PARINIT   0x00000001
#define EF_S390_HIGH_GPRS       0x00000001
#define DF_1_GLOBAL     0x00000002
#define DF_P1_GROUPPERM 0x00000002
#define DF_SYMBOLIC     0x00000002      /* Symbol resolutions start
with this object */
#define DTF_1_CONFEXP   0x00000002
#define DF_1_GROUP      0x00000004
#define DF_TEXTREL        0x00000004        /* Object contains text
relocations */
#define DF_1_NODELETE   0x00000008
#define DF_BIND_NOW     0x00000008      /* No lazy binding for this
object */
#define EF_IA_64_MASKOS 0x0000000f
#define DF_1_LOADFLTR   0x00000010
#define DF_STATIC_TLS    0x00000010       /* Module uses the static
TLS model */
#define EF_IA_64_ABI64  0x00000010
#define DF_1_INITFIRST  0x00000020
#define DF_1_NOOPEN     0x00000040
```

```
#define DF_1_ORIGIN     0x00000080
#define DF_1_DIRECT     0x00000100
#define DF_1_TRANS      0x00000200
#define DF_1_INTERPOSE  0x00000400
#define DF_1_NODEFLIB   0x00000800
#define DF_1_NODUMP     0x00001000
#define DF_1_CONFALT    0x00002000
#define DF_1_ENDFILTEE  0x00004000
#define DF_1_DISPRELDNE 0x00008000
#define SYMINFO_FLG_DIRECT     0x0001
#define DF_1_DISPRELPND 0x00010000
#define SYMINFO_FLG_PASSTHRU    0x0002
#define SYMINFO_FLG_COPY       0x0004
#define SYMINFO_FLG_LAZYLOAD    0x0008
#define EF_CPU32       0x00810000
#define PF_MASKOS      0x0ff00000
#define SHF_MASKOS     0x0ff00000
#define GRP_COMDAT     0x1
#define SHF_IA_64_SHORT 0x10000000
#define SHF_IA_64_NORECOV      0x20000000
#define R_IA64_IMM14    0x21
#define R_IA64_IMM22    0x22
#define R_IA64_IMM64    0x23
#define R_IA64_DIR32MSB 0x24
#define R_IA64_DIR32LSB 0x25
#define R_IA64_DIR64MSB 0x26
#define R_IA64_DIR64LSB 0x27
#define R_IA64_GPREL22  0x2a
#define R_IA64_GPREL64I 0x2b
#define R_IA64_GPREL32MSB      0x2c
#define R_IA64_GPREL32LSB      0x2d
#define R_IA64_GPREL64MSB      0x2e
#define R_IA64_GPREL64LSB      0x2f
#define R_IA64_LTOFF22  0x32
#define R_IA64_LTOFF64I 0x33
#define R_IA64_PLTOFF22 0x3a
#define R_IA64_PLTOFF64I       0x3b
#define R_IA64_PLTOFF64MSB     0x3e
#define R_IA64_PLTOFF64LSB     0x3f
#define R_IA64_FPTR64I  0x43
#define R_IA64_FPTR32MSB       0x44
#define R_IA64_FPTR32LSB       0x45
#define R_IA64_FPTR64MSB       0x46
#define R_IA64_FPTR64LSB       0x47
#define R_IA64_PCREL60B 0x48
#define R_IA64_PCREL21B 0x49
#define R_IA64_PCREL21M 0x4a
#define R_IA64_PCREL21F 0x4b
#define R_IA64_PCREL32MSB      0x4c
#define R_IA64_PCREL32LSB      0x4d
#define R_IA64_PCREL64MSB      0x4e
#define R_IA64_PCREL64LSB      0x4f
#define R_IA64_LTOFF_FPTR22    0x52
#define R_IA64_LTOFF_FPTR64I    0x53
#define R_IA64_LTOFF_FPTR32MSB  0x54
#define R_IA64_LTOFF_FPTR32LSB  0x55
#define R_IA64_LTOFF_FPTR64MSB  0x56
#define R_IA64_LTOFF_FPTR64LSB  0x57
#define R_IA64_SEGREL32MSB     0x5c
#define R_IA64_SEGREL32LSB     0x5d
#define R_IA64_SEGREL64MSB     0x5e
#define R_IA64_SEGREL64LSB     0x5f
#define PT_LOOS 0x60000000
#define SHT_LOOS       0x60000000
#define DT_LOOS 0x6000000d
#define R_IA64_SECREL32MSB     0x64
```



```
#define PT_GNU_EH_FRAME 0x6474e550
#define PT_GNU_STACK    0x6474e551
#define PT_GNU_RELRO    0x6474e552
#define R_IA64_SECREL32LSB     0x65
#define R_IA64_SECREL64MSB     0x66
#define R_IA64_SECREL64LSB     0x67
#define R_IA64_REL32MSB 0x6c
#define R_IA64_REL32LSB 0x6d
#define R_IA64_REL64MSB 0x6e
#define R_IA64_REL64LSB 0x6f
#define DT_HIOS 0x6ffff000
#define DT_VALRNGLO    0x6ffffd00
#define DT_GNU_PRELINKED       0x6ffffdf5
#define DT_GNU_CONFLICTSZ      0x6ffffdf6
#define DT_GNU_LIBLISTSZ       0x6ffffdf7
#define DT_CHECKSUM    0x6ffffdf8
#define DT_PLTPADSZ    0x6ffffdf9
#define DT_MOVEENT     0x6ffffdfa
#define DT_MOVESZ      0x6ffffdfb
#define DT_FEATURE_1   0x6ffffdfc
#define DT_POSFLAG_1   0x6ffffdfd
#define DT_SYMINSZ     0x6ffffdfe
#define DT_SYMINENT    0x6ffffdff
#define DT_VALRNGHI    0x6ffffdff
#define DT_ADDRRNGLO   0x6ffffe00
#define DT_GNU_HASH    0x6ffffef5
#define DT_TLSDESC_PLT 0x6ffffef6
#define DT_TLSDESC_GOT 0x6ffffef7
#define DT_GNU_CONFLICT 0x6ffffef8
#define DT_GNU_LIBLIST 0x6ffffef9
#define DT_CONFIG      0x6ffffefa
#define DT_DEPAUDIT    0x6ffffefb
#define DT_AUDIT       0x6ffffefc
#define DT_PLTPAD      0x6ffffefd
#define DT_MOVETAB     0x6ffffefe
#define DT_ADDRRNGHI   0x6fffeff
#define DT_SYMINFO     0x6ffffeff
#define DT_VERSYM      0x6fffffff0
#define SHT_GNU_ATTRIBUTES     0x6ffffff5
#define SHT_GNU_HASH   0x6ffffff6
#define SHT_GNU_LIBLIST 0x6ffffff7
#define SHT_CHECKSUM   0x6ffffff8
#define DT_RELACOUNT   0x6ffffff9
#define DT_RELCOUNT    0x6ffffffa
#define DT_FLAGS_1     0x6ffffffb
#define DT_VERDEF      0x6ffffffc
#define DT_VERDEFNUM   0x6ffffffd
#define SHT_GNU_verdef  0x6ffffffd
#define DT_VERNEED     0x6ffffffe
#define SHT_GNU_verneed 0x6ffffffe
#define DT_VERNEEDNUM   0x6fffffff
#define PT_HIOS 0x6fffffff
#define SHT_GNU_versym  0x6fffffff
#define SHT_HIOS       0x6fffffff
#define DT_LOPROC      0x70000000
#define PT_LOPROC      0x70000000
#define SHT_LOPROC     0x70000000
#define R_IA64_LTV32MSB 0x74
#define R_IA64_LTV32LSB 0x75
#define R_IA64_LTV64MSB 0x76
#define R_IA64_LTV64LSB 0x77
#define R_IA64_PCREL21BI       0x79
#define R_IA64_PCREL22  0x7a
#define R_IA64_PCREL64I 0x7b
#define ELFMAG0 0x7f
#define DT_AUXILIARY   0x7ffffffd
```

```
#define DT_FILTER      0x7fffffff
#define DT_HIPROC      0x7fffffff
#define PT_HIPROC      0x7fffffff
#define SHT_HIPROC     0x7fffffff
#define R_IA64_IPLTMSB  0x80
#define PF_IA_64_NORECOV       0x80000000
#define SHT_LOUSER     0x80000000
#define R_IA64_IPLTLSB  0x81
#define R_IA64_COPY    0x84
#define R_IA64_SUB     0x85
#define R_IA64_LTOFF22X 0x86
#define R_IA64_LDXMOV   0x87
#define SHT_HIUSER     0x8fffffff
#define R_IA64_TPREL14  0x91
#define R_IA64_TPREL22  0x92
#define R_IA64_TPREL64I 0x93
#define R_IA64_TPREL64MSB      0x96
#define R_IA64_TPREL64LSB      0x97
#define R_IA64_LTOFF_TPREL22    0x9a
#define R_IA64_DTPMOD64MSB     0xa6
#define R_IA64_DTPMOD64LSB     0xa7
#define R_IA64_LTOFF_DTPMOD22   0xaa
#define R_IA64_DTPREL14 0xb1
#define R_IA64_DTPREL22 0xb2
#define R_IA64_DTPREL64I       0xb3
#define R_IA64_DTPREL32MSB     0xb4
#define R_IA64_DTPREL32LSB     0xb5
#define R_IA64_DTPREL64MSB     0xb6
#define R_IA64_DTPREL64LSB     0xb7
#define R_IA64_LTOFF_DTPREL22   0xba
#define PF_MASKPROC    0xf0000000
#define SHF_MASKPROC    0xf0000000
#define ET_LOOS 0xfe00
#define ET_HIOS 0xfeff
#define ET_LOPROC      0xff00
#define SHN_LOPROC     0xff00
#define SHN_LORESERVE   0xff00
#define SYMINFO_BT_LOWRESERVE   0xff00
#define EF_IA_64_ARCH   0xff000000
#define SHN_HIPROC     0xff1f
#define SHN_LOOS       0xff20
#define SHN_HIOS       0xff3f
#define SHN_ABS 0xfff1
#define SHN_COMMON     0xfff2
#define SYMINFO_BT_PARENT      0xfffe
#define ET_HIPROC      0xffff
#define PN_XNUM 0xffff
#define SHN_HIRESERVE   0xffff
#define SHN_XINDEX     0xffff
#define SYMINFO_BT_SELF 0xffff
#define DT_IA_64_NUM    1
#define DT_NEEDED      1
#define DT_PPC_NUM     1
#define EI_MAG1 1
#define ELFCLASS32     1
#define ELFDATA2LSB    1
#define ELF_NOTE_OS_GNU 1
#define ET_REL  1
#define EV_CURRENT     1
#define NT_GNU_ABI_TAG  1
#define PT_LOAD 1
#define R_386_32       1
#define R_390_8 1
#define R_PPC_ADDR32    1
#define R_X86_64_64    1
#define SHT_PROGBITS    1
```



```
#define STB_GLOBAL     1
#define STT_OBJECT     1
#define STV_INTERNAL   1
#define SYMINFO_CURRENT 1
#define DT_STRSZ       10
#define R_386_GOTPC    10
#define R_390_GLOB_DAT 10
#define R_PPC_REL24    10
#define R_X86_64_32    10
#define SHT_SHLIB      10
#define STB_LOOS       10
#define STT_GNU_IFUNC  10
#define STT_LOOS       10
#define R_PPC64_TPREL16_HIGHESTA       100
#define R_PPC64_DTPREL16_DS    101
#define R_PPC64_DTPREL16_LO_DS 102
#define R_PPC64_DTPREL16_HIGHER 103
#define R_PPC64_DTPREL16_HIGHERA       104
#define R_PPC64_DTPREL16_HIGHEST       105
#define R_PPC64_DTPREL16_HIGHESTA      106
#define DT_ADDRNUM     11
#define DT_SYMENT      11
#define R_386_32PLT    11
#define R_390_JMP_SLOT 11
#define R_PPC_REL14    11
#define R_X86_64_32S   11
#define SHT_DYNSYM     11
#define DT_INIT 12
#define DT_VALNUM      12
#define R_390_RELATIVE 12
#define R_PPC_REL14_BRTAKEN    12
#define R_X86_64_16    12
#define STB_HIOS       12
#define STT_HIOS       12
#define DT_FINI 13
#define R_390_GOTOFF32 13
#define R_PPC_REL14_BRNTAKEN   13
#define R_X86_64_PC16  13
#define STB_LOPROC     13
#define STT_LOPROC     13
#define DT_SONAME      14
#define R_386_TLS_TPOFF 14
#define R_390_GOTPC    14
#define R_PPC_GOT16    14
#define R_X86_64_8     14
#define SHT_INIT_ARRAY 14
#define DT_RPATH       15
#define R_386_TLS_IE   15
#define R_390_GOT16    15
#define R_PPC_GOT16_LO 15
#define R_X86_64_PC8   15
#define SHT_FINI_ARRAY 15
#define STB_HIPROC     15
#define STT_HIPROC     15
#define DT_SYMBOLIC    16
#define DT_VERSIONTAGNUM       16
#define R_386_TLS_GOTIE 16
#define R_390_PC16     16
#define R_PPC_GOT16_HI 16
#define R_X86_64_DTPMOD64      16
#define SHT_PREINIT_ARRAY      16
#define DT_REL 17
#define R_386_TLS_LE   17
#define R_390_PC16DBL  17
#define R_PPC_GOT16_HA 17
#define R_X86_64_DTPOFF64      17
```

```
#define SHT_GROUP      17
#define DT_RELSZ       18
#define R_386_TLS_GD    18
#define R_390_PLT16DBL  18
#define R_PPC_PLTREL24  18
#define R_X86_64_TPOFF64       18
#define SHT_SYMTAB_SHNDX       18
#define DT_RELENT      19
#define R_386_TLS_LDM   19
#define R_390_PC32DBL   19
#define R_PPC_COPY     19
#define R_X86_64_TLSGD  19
#define DT_PLTRELSZ    2
#define EI_MAG2 2
#define ELFCLASS64     2
#define ELFDATA2MSB    2
#define ELF_NOTE_OS_SOLARIS2    2
#define ET_EXEC 2
#define EV_NUM  2
#define PT_DYNAMIC     2
#define R_386_PC32     2
#define R_390_12       2
#define R_PPC_ADDR24    2
#define R_X86_64_PC32   2
#define SHT_SYMTAB     2
#define STB_WEAK       2
#define STT_FUNC       2
#define STV_HIDDEN     2
#define SYMINFO_NUM    2
#define DT_PLTREL      20
#define EM_PPC  20
#define R_386_16       20
#define R_390_PLT32DBL  20
#define R_PPC_GLOB_DAT  20
#define R_X86_64_TLSLD  20
#define DT_DEBUG       21
#define EM_PPC64       21
#define R_386_PC16     21
#define R_390_GOTPCDBL  21
#define R_PPC_JMP_SLOT  21
#define R_X86_64_DTPOFF32      21
#define DT_TEXTREL     22
#define EM_S390 22
#define R_386_8 22
#define R_390_64       22
#define R_PPC_RELATIVE  22
#define R_X86_64_GOTTPOFF      22
#define DT_JMPREL      23
#define R_386_PC8      23
#define R_390_PC64     23
#define R_PPC_LOCAL24PC 23
#define R_X86_64_TPOFF32       23
#define DT_BIND_NOW    24
#define R_386_TLS_GD_32 24
#define R_390_GOT64    24
#define R_PPC_UADDR32   24
#define R_X86_64_PC64   24
#define R_PPC64_JMP_IREL       247
#define R_PPC64_IRELATIVE      248
#define R_PPC_IRELATIVE 248
#define R_PPC64_REL16   249
#define R_PPC_REL16    249
#define DT_INIT_ARRAY   25
#define R_386_TLS_GD_PUSH      25
#define R_390_PLT64    25
#define R_PPC_UADDR16   25
```



```
#define R_X86_64_GOTOFF64       25
#define R_PPC64_REL16_LO        250
#define R_PPC_REL16_LO  250
#define R_PPC64_REL16_HI        251
#define R_PPC_REL16_HI  251
#define R_PPC64_REL16_HA        252
#define R_PPC_REL16_HA  252
#define R_PPC_TOC16     255
#define DT_FINI_ARRAY   26
#define R_386_TLS_GD_CALL       26
#define R_390_GOTENT    26
#define R_PPC_REL32     26
#define R_X86_64_GOTPC32        26
#define DT_INIT_ARRAYSZ 27
#define R_386_TLS_GD_POP        27
#define R_390_GOTOFF16  27
#define R_PPC_PLT32     27
#define R_X86_64_GOT64  27
#define DT_FINI_ARRAYSZ 28
#define R_386_TLS_LDM_32        28
#define R_390_GOTOFF64  28
#define R_PPC_PLTREL32  28
#define R_X86_64_GOTPCREL64     28
#define DT_RUNPATH      29
#define R_386_TLS_LDM_PUSH      29
#define R_390_GOTPLT12  29
#define R_PPC_PLT16_LO  29
#define R_X86_64_GOTPC64        29
#define DT_EXTRANUM     3
#define DT_PLTGOT       3
#define DT_PPC64_NUM    3
#define EI_MAG3 3
#define ELFCLASSNUM     3
#define ELFDATANUM      3
#define ELFOSABI_LINUX  3
#define ELF_NOTE_OS_FREEBSD     3
#define EM_386  3
#define ET_DYN  3
#define PT_INTERP       3
#define R_386_GOT32     3
#define R_390_16        3
#define R_PPC_ADDR16    3
#define R_X86_64_GOT32  3
#define SHT_STRTAB      3
#define STB_NUM 3
#define STT_SECTION     3
#define STV_PROTECTED   3
#define DT_FLAGS        30
#define R_386_TLS_LDM_CALL      30
#define R_390_GOTPLT16  30
#define R_PPC_PLT16_HI  30
#define R_X86_64_GOTPLT64       30
#define R_386_TLS_LDM_POP       31
#define R_390_GOTPLT32  31
#define R_PPC_PLT16_HA  31
#define R_X86_64_PLTOFF64       31
#define DT_ENCODING     32
#define DT_PREINIT_ARRAY        32
#define R_386_TLS_LDO_32        32
#define R_390_GOTPLT64  32
#define R_PPC_SDAREL16  32
#define R_X86_64_SIZE32 32
#define DT_PREINIT_ARRAYSZ      33
#define R_386_TLS_IE_32 33
#define R_390_GOTPLTENT 33
#define R_PPC_SECTOFF   33
```

```
#define R_X86_64_SIZE64 33
#define DT_NUM  34
#define R_386_TLS_LE_32 34
#define R_390_PLTOFF16  34
#define R_PPC_SECTOFF_LO        34
#define R_X86_64_GOTPC32_TLSDESC        34
#define R_386_TLS_DTPMOD32      35
#define R_390_PLTOFF32  35
#define R_PPC_SECTOFF_HI        35
#define R_X86_64_TLSDESC_CALL   35
#define R_386_TLS_DTPOFF32      36
#define R_390_PLTOFF64  36
#define R_PPC_SECTOFF_HA        36
#define R_X86_64_TLSDESC        36
#define R_386_TLS_TPOFF32       37
#define R_390_TLS_LOAD  37
#define R_PPC64_ADDR30  37
#define R_X86_64_IRELATIVE      37
#define R_390_TLS_GDCALL        38
#define R_PPC64_ADDR64  38
#define R_X86_64_NUM    38
#define R_386_TLS_GOTDESC       39
#define R_390_TLS_LDCALL        39
#define R_PPC64_ADDR16_HIGHER   39
#define DT_HASH 4
#define EI_CLASS        4
#define ET_CORE 4
#define PT_NOTE 4
#define R_386_PLT32     4
#define R_390_32        4
#define R_PPC_ADDR16_LO 4
#define R_X86_64_PLT32  4
#define SELFMAG 4
#define SHT_RELA        4
#define STT_FILE        4
#define EM_ARM  40
#define R_386_TLS_DESC_CALL     40
#define R_390_TLS_GD32  40
#define R_PPC64_ADDR16_HIGHERA  40
#define R_386_TLS_DESC  41
#define R_390_TLS_GD64  41
#define R_PPC64_ADDR16_HIGHEST  41
#define R_386_IRELATIVE 42
#define R_390_TLS_GOTIE12       42
#define R_PPC64_ADDR16_HIGHESTA 42
#define R_386_NUM       43
#define R_390_TLS_GOTIE32       43
#define R_PPC64_UADDR64 43
#define R_390_TLS_GOTIE64       44
#define R_PPC64_REL64   44
#define R_390_TLS_LDM32 45
#define R_PPC64_PLT64   45
#define R_390_TLS_LDM64 46
#define R_PPC64_PLTREL64        46
#define R_390_TLS_IE32  47
#define R_PPC64_TOC16   47
#define R_390_TLS_IE64  48
#define R_PPC64_TOC16_LO        48
#define R_390_TLS_IEENT 49
#define R_PPC64_TOC16_HI        49
#define DT_STRTAB       5
#define EI_DATA 5
#define ET_NUM  5
#define PT_SHLIB        5
#define R_386_COPY      5
#define R_390_PC32      5
```



```
#define R_PPC_ADDR16_HI 5
#define R_X86_64_COPY   5
#define SHT_HASH        5
#define STT_COMMON      5
#define EM_IA_64        50
#define R_390_TLS_LE32  50
#define R_PPC64_TOC16_HA        50
#define R_390_TLS_LE64  51
#define R_PPC64_TOC     51
#define R_390_TLS_LDO32 52
#define R_PPC64_PLTGOT16        52
#define R_390_TLS_LDO64 53
#define R_PPC64_PLTGOT16_LO     53
#define R_390_TLS_DTPMOD        54
#define R_PPC64_PLTGOT16_HI     54
#define R_390_TLS_DTPOFF        55
#define R_PPC64_PLTGOT16_HA     55
#define R_390_TLS_TPOFF 56
#define R_PPC64_ADDR16_DS       56
#define R_390_20        57
#define R_PPC64_ADDR16_LO_DS    57
#define R_390_GOT20     58
#define R_PPC64_GOT16_DS        58
#define R_390_GOTPLT20  59
#define R_PPC64_GOT16_LO_DS     59
#define DT_SYMTAB       6
#define EI_VERSION      6
#define PT_PHDR 6
#define R_386_GLOB_DAT  6
#define R_390_GOT12     6
#define R_PPC_ADDR16_HA 6
#define R_X86_64_GLOB_DAT       6
#define SHT_DYNAMIC     6
#define STT_TLS 6
#define R_390_TLS_GOTIE20       60
#define R_PPC64_PLT16_LO_DS     60
#define R_390_NUM       61
#define R_PPC64_SECTOFF_DS      61
#define EM_X86_64       62
#define R_PPC64_SECTOFF_LO_DS   62
#define R_PPC64_TOC16_DS        63
#define R_PPC64_TOC16_LO_DS     64
#define R_PPC64_PLTGOT16_DS     65
#define R_PPC64_PLTGOT16_LO_DS  66
#define R_PPC64_TLS     67
#define R_PPC_TLS       67
#define R_PPC64_DTPMOD64        68
#define R_PPC_DTPMOD32  68
#define R_PPC64_TPREL16 69
#define R_PPC_TPREL16   69
#define DT_RELA 7
#define EI_OSABI        7
#define PT_TLS  7
#define R_386_JMP_SLOT  7
#define R_390_GOT32     7
#define R_PPC_ADDR14    7
#define R_X86_64_JUMP_SLOT      7
#define SHT_NOTE        7
#define STT_NUM 7
#define R_PPC64_TPREL16_LO      70
#define R_PPC_TPREL16_LO        70
#define R_PPC64_TPREL16_HI      71
#define R_PPC_TPREL16_HI        71
#define R_PPC64_TPREL16_HA      72
#define R_PPC_TPREL16_HA        72
#define R_PPC64_TPREL64 73
```

```
#define R_PPC_TPREL32   73
#define R_PPC64_DTPREL16       74
#define R_PPC_DTPREL16  74
#define R_PPC64_DTPREL16_LO    75
#define R_PPC_DTPREL16_LO      75
#define R_PPC64_DTPREL16_HI    76
#define R_PPC_DTPREL16_HI      76
#define R_PPC64_DTPREL16_HA    77
#define R_PPC_DTPREL16_HA      77
#define R_PPC64_DTPREL64       78
#define R_PPC_DTPREL32  78
#define R_PPC64_GOT_TLSGD16    79
#define R_PPC_GOT_TLSGD16      79
#define DT_RELASZ      8
#define EI_ABIVERSION   8
#define PT_NUM  8
#define R_386_RELATIVE  8
#define R_390_PLT32    8
#define R_PPC_ADDR14_BRTAKEN    8
#define R_X86_64_RELATIVE      8
#define SHT_NOBITS     8
#define R_PPC64_GOT_TLSGD16_LO  80
#define R_PPC_GOT_TLSGD16_LO    80
#define R_PPC64_GOT_TLSGD16_HI  81
#define R_PPC_GOT_TLSGD16_HI    81
#define R_PPC64_GOT_TLSGD16_HA  82
#define R_PPC_GOT_TLSGD16_HA    82
#define R_PPC64_GOT_TLSLD16    83
#define R_PPC_GOT_TLSLD16      83
#define R_PPC64_GOT_TLSLD16_LO  84
#define R_PPC_GOT_TLSLD16_LO    84
#define R_PPC64_GOT_TLSLD16_HI  85
#define R_PPC_GOT_TLSLD16_HI    85
#define R_PPC64_GOT_TLSLD16_HA  86
#define R_PPC_GOT_TLSLD16_HA    86
#define R_PPC64_GOT_TPREL16_DS  87
#define R_PPC_GOT_TPREL16    87
#define R_PPC64_GOT_TPREL16_LO_DS      88
#define R_PPC_GOT_TPREL16_LO    88
#define R_PPC64_GOT_TPREL16_HI  89
#define R_PPC_GOT_TPREL16_HI    89
#define DT_RELAENT     9
#define EI_PAD  9
#define R_386_GOTOFF    9
#define R_390_COPY     9
#define R_PPC_ADDR14_BRNTAKEN   9
#define R_X86_64_GOTPCREL      9
#define SHT_REL 9
#define R_PPC64_GOT_TPREL16_HA  90
#define R_PPC_GOT_TPREL16_HA    90
#define R_PPC64_GOT_DTPREL16_DS 91
#define R_PPC_GOT_DTPREL16     91
#define R_PPC64_GOT_DTPREL16_LO_DS     92
#define R_PPC_GOT_DTPREL16_LO   92
#define R_PPC64_GOT_DTPREL16_HI 93
#define R_PPC_GOT_DTPREL16_HI   93
#define R_PPC64_GOT_DTPREL16_HA 94
#define R_PPC_GOT_DTPREL16_HA   94
#define R_PPC64_TPREL16_DS     95
#define R_PPC64_TPREL16_LO_DS   96
#define R_PPC64_TPREL16_HIGHER  97
#define R_PPC64_TPREL16_HIGHERA 98
#define R_PPC64_TPREL16_HIGHEST 99
#define ELF64_M_INFO(sym, size) ELF32_M_INFO (sym, size)
#define ELF64_M_SIZE(info)     ELF32_M_SIZE (info)
#define ELF64_M_SYM(info)      ELF32_M_SYM (info)
```



```
#define ELF64_ST_BIND(val)      ELF32_ST_BIND (val)
#define  ELF64_ST_INFO(bind,type)           ELF32_ST_INFO  ((bind),
(type))
#define ELF64_ST_TYPE(val)      ELF32_ST_TYPE (val)
#define ELF64_ST_VISIBILITY(o)  ELF32_ST_VISIBILITY(o)
#define ELF_NOTE_GNU    "GNU"
#define ELF_NOTE_ABI    NT_GNU_ABI_TAG
#define R_PPC64_ADDR14  R_PPC_ADDR14
#define R_PPC64_ADDR14_BRNTAKEN R_PPC_ADDR14_BRNTAKEN
#define R_PPC64_ADDR14_BRTAKEN  R_PPC_ADDR14_BRTAKEN
#define R_PPC64_ADDR16  R_PPC_ADDR16
#define R_PPC64_ADDR16_HA       R_PPC_ADDR16_HA
#define R_PPC64_ADDR16_HI       R_PPC_ADDR16_HI
#define R_PPC64_ADDR16_LO       R_PPC_ADDR16_LO
#define R_PPC64_ADDR24  R_PPC_ADDR24
#define R_PPC64_ADDR32  R_PPC_ADDR32
#define R_PPC64_COPY    R_PPC_COPY
#define R_PPC64_GLOB_DAT        R_PPC_GLOB_DAT
#define R_PPC64_GOT16   R_PPC_GOT16
#define R_PPC64_GOT16_HA        R_PPC_GOT16_HA
#define R_PPC64_GOT16_HI        R_PPC_GOT16_HI
#define R_PPC64_GOT16_LO        R_PPC_GOT16_LO
#define R_PPC64_JMP_SLOT        R_PPC_JMP_SLOT
#define R_PPC64_NONE    R_PPC_NONE
#define R_PPC64_PLT16_HA        R_PPC_PLT16_HA
#define R_PPC64_PLT16_HI        R_PPC_PLT16_HI
#define R_PPC64_PLT16_LO        R_PPC_PLT16_LO
#define R_PPC64_PLT32   R_PPC_PLT32
#define R_PPC64_PLTREL32        R_PPC_PLTREL32
#define R_PPC64_REL14   R_PPC_REL14
#define R_PPC64_REL14_BRNTAKEN  R_PPC_REL14_BRNTAKEN
#define R_PPC64_REL14_BRTAKEN   R_PPC_REL14_BRTAKEN
#define R_PPC64_REL24   R_PPC_REL24
#define R_PPC64_REL32   R_PPC_REL32
#define R_PPC64_RELATIVE        R_PPC_RELATIVE
#define R_PPC64_SECTOFF R_PPC_SECTOFF
#define R_PPC64_SECTOFF_HA      R_PPC_SECTOFF_HA
#define R_PPC64_SECTOFF_HI      R_PPC_SECTOFF_HI
#define R_PPC64_SECTOFF_LO      R_PPC_SECTOFF_LO
#define R_PPC64_UADDR16 R_PPC_UADDR16
#define R_PPC64_UADDR32 R_PPC_UADDR32
#define ELFMAG  "\177ELF"

typedef uint32_t Elf32_Addr;
typedef uint64_t Elf64_Addr;
typedef uint32_t Elf32_Word;
typedef uint32_t Elf64_Word;
typedef int32_t Elf32_Sword;
typedef int32_t Elf64_Sword;
typedef uint64_t Elf32_Xword;
typedef uint64_t Elf64_Xword;
typedef int64_t Elf32_Sxword;
typedef int64_t Elf64_Sxword;
typedef uint32_t Elf32_Off;
typedef uint64_t Elf64_Off;
typedef struct {
    Elf32_Word p_type;          /* Segment type */
    Elf32_Off p_offset;         /* Segment file offset */
    Elf32_Addr p_vaddr;         /* Segment virtual address */
    Elf32_Addr p_paddr;         /* Segment physical address */
    Elf32_Word p_filesz;        /* Segment size in file */
    Elf32_Word p_memsz;         /* Segment size in memory */
    Elf32_Word p_flags;         /* Segment flags */
    Elf32_Word p_align;         /* Segment alignment */
} Elf32_Phdr;
typedef struct {
```

```
    Elf64_Word p_type;          /* Segment type */
    Elf64_Word p_flags;         /* Segment flags */
    Elf64_Off p_offset;         /* Segment file offset */
    Elf64_Addr p_vaddr;         /* Segment virtual address */
    Elf64_Addr p_paddr;         /* Segment physical address */
    Elf64_Xword p_filesz;       /* Segment size in file */
    Elf64_Xword p_memsz;        /* Segment size in memory */
    Elf64_Xword p_align;        /* Segment alignment */
} Elf64_Phdr;
typedef uint16_t Elf32_Half;
typedef uint16_t Elf64_Half;
typedef uint16_t Elf32_Section;
typedef uint16_t Elf64_Section;
typedef struct {
    Elf32_Word n_namesz;
    Elf32_Word n_descsz;
    Elf32_Word n_type;
} Elf32_Nhdr;
typedef struct {
    Elf64_Word n_namesz;
    Elf64_Word n_descsz;
    Elf64_Word n_type;
} Elf64_Nhdr;
typedef struct {
    Elf64_Word st_name;
    unsigned char st_info;
    unsigned char st_other;
    Elf64_Section st_shndx;
    Elf64_Addr st_value;
    Elf64_Xword st_size;
} Elf64_Sym;
typedef struct {
    Elf32_Word st_name;
    Elf32_Addr st_value;
    Elf32_Word st_size;
    unsigned char st_info;
    unsigned char st_other;
    Elf32_Section st_shndx;
} Elf32_Sym;
typedef struct {
    Elf64_Addr r_offset;
    Elf64_Xword r_info;
} Elf64_Rel;
typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
} Elf32_Rel;
typedef struct {
    Elf64_Addr r_offset;
    Elf64_Xword r_info;
    Elf64_Sxword r_addend;
} Elf64_Rela;
typedef struct {
    Elf32_Addr r_offset;
    Elf32_Word r_info;
    Elf32_Sword r_addend;
} Elf32_Rela;
typedef struct {
    Elf32_Half vd_version;
    Elf32_Half vd_flags;
    Elf32_Half vd_ndx;
    Elf32_Half vd_cnt;
    Elf32_Word vd_hash;
    Elf32_Word vd_aux;
    Elf32_Word vd_next;
} Elf32_Verdef;
```



```
typedef struct {
    Elf64_Half vd_version;
    Elf64_Half vd_flags;
    Elf64_Half vd_ndx;
    Elf64_Half vd_cnt;
    Elf64_Word vd_hash;
    Elf64_Word vd_aux;
    Elf64_Word vd_next;
} Elf64_Verdef;
typedef struct {
    Elf64_Word vda_name;
    Elf64_Word vda_next;
} Elf64_Verdaux;
typedef struct {
    Elf32_Word vda_name;
    Elf32_Word vda_next;
} Elf32_Verdaux;
typedef struct {
    Elf32_Half vn_version;
    Elf32_Half vn_cnt;
    Elf32_Word vn_file;
    Elf32_Word vn_aux;
    Elf32_Word vn_next;
} Elf32_Verneed;
typedef struct {
    Elf64_Half vn_version;
    Elf64_Half vn_cnt;
    Elf64_Word vn_file;
    Elf64_Word vn_aux;
    Elf64_Word vn_next;
} Elf64_Verneed;
typedef struct {
    Elf32_Word vna_hash;
    Elf32_Half vna_flags;
    Elf32_Half vna_other;
    Elf32_Word vna_name;
    Elf32_Word vna_next;
} Elf32_Vernaux;
typedef struct {
    Elf64_Word vna_hash;
    Elf64_Half vna_flags;
    Elf64_Half vna_other;
    Elf64_Word vna_name;
    Elf64_Word vna_next;
} Elf64_Vernaux;
typedef struct {
    unsigned char e_ident[EI_NIDENT];
    Elf64_Half e_type;
    Elf64_Half e_machine;
    Elf64_Word e_version;
    Elf64_Addr e_entry;
    Elf64_Off e_phoff;
    Elf64_Off e_shoff;
    Elf64_Word e_flags;
    Elf64_Half e_ehsize;
    Elf64_Half e_phentsize;
    Elf64_Half e_phnum;
    Elf64_Half e_shentsize;
    Elf64_Half e_shnum;
    Elf64_Half e_shstrndx;
} Elf64_Ehdr;
typedef struct {
    unsigned char e_ident[EI_NIDENT];
    Elf32_Half e_type;
    Elf32_Half e_machine;
    Elf32_Word e_version;
```

```
    Elf32_Addr e_entry;
    Elf32_Off e_phoff;
    Elf32_Off e_shoff;
    Elf32_Word e_flags;
    Elf32_Half e_ehsize;
    Elf32_Half e_phentsize;
    Elf32_Half e_phnum;
    Elf32_Half e_shentsize;
    Elf32_Half e_shnum;
    Elf32_Half e_shstrndx;
} Elf32_Ehdr;
typedef struct {
    Elf32_Word sh_name;
    Elf32_Word sh_type;
    Elf32_Word sh_flags;
    Elf32_Addr sh_addr;
    Elf32_Off sh_offset;
    Elf32_Word sh_size;
    Elf32_Word sh_link;
    Elf32_Word sh_info;
    Elf32_Word sh_addralign;
    Elf32_Word sh_entsize;
} Elf32_Shdr;
typedef struct {
    Elf64_Word sh_name;
    Elf64_Word sh_type;
    Elf64_Xword sh_flags;
    Elf64_Addr sh_addr;
    Elf64_Off sh_offset;
    Elf64_Xword sh_size;
    Elf64_Word sh_link;
    Elf64_Word sh_info;
    Elf64_Xword sh_addralign;
    Elf64_Xword sh_entsize;
} Elf64_Shdr;
typedef struct {
    Elf32_Sword d_tag;
    union {
        Elf32_Word d_val;
        Elf32_Addr d_ptr;
    } d_un;
} Elf32_Dyn;
typedef struct {
    Elf64_Sxword d_tag;
    union {
        Elf64_Xword d_val;
        Elf64_Addr d_ptr;
    } d_un;
} Elf64_Dyn;
```

14.4.8 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
```

14.4.9 envz.h

```
extern error_t envz_add(char **envz, size_t * envz_len, const char
*name,
                        const char *value);
```



```
extern char envz_entry(const char *envz, size_t envz_len,
                       const char *name);
extern char envz_get(const char *envz, size_t envz_len, const char
*name);
extern error_t envz_merge(char **envz, size_t * envz_len,
                          const char *envz2, size_t envz2_len,
                          int override);
extern void envz_remove(char **envz, size_t * envz_len, const char
*name);
extern void envz_strip(char **envz, size_t * envz_len);
```

14.4.10 err.h

```
extern void err(int eval, const char *fmt, ...);
extern void errx(int eval, const char *fmt, ...);
extern void verrx(int eval, const char *fmt, va_list args);
extern void warn(const char *fmt, ...);
extern void warnx(const char *fmt, ...);
```

14.4.11 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 ETOOMANYREFS   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 ESTALE  116          /* Stale NFS file handle */
#define EUCLEAN 117          /* Structure needs cleaning */
#define ENOTNAM 118          /* Not a XENIX named type file */
#define ENAVAIL 119          /* No XENIX semaphores available */
#define ENOMEM  12           /* Out of memory */
#define EISNAM  120          /* Is a named type file */
#define EREMOTEIO      121   /* Remote I/O error */
#define EDQUOT  122          /* Quota exceeded */
#define ENOMEDIUM      123   /* No medium found */
#define EMEDIUMTYPE    124   /* Wrong medium type */
#define ECANCELED      125   /* Operation Canceled */
#define EACCES  13           /* Permission denied */
#define EOWNERDEAD     130   /* Owner died */
#define ENOTRECOVERABLE 131   /* State not recoverable */
```

```
#define ERFKILL 132          /* Operation not possible due to RF-
kill */
#define EFAULT  14           /* Bad address */
#define ENOTBLK 15           /* Block device required */
#define EBUSY   16           /* Device or resource busy */
#define EEXIST  17           /* File exists */
#define EXDEV   18           /* Cross-device link */
#define ENODEV  19           /* No such device */
#define ENOENT  2            /* No such file or directory */
#define ENOTDIR 20           /* Not a directory */
#define EISDIR  21           /* Is a directory */
#define EINVAL  22           /* Invalid argument */
#define ENFILE  23           /* File table overflow */
#define EMFILE  24           /* Too many open files */
#define ENOTTY  25           /* Not a typewriter */
#define ETXTBSY 26           /* Text file busy */
#define EFBIG   27           /* File too large */
#define ENOSPC  28           /* No space left on device */
#define ESPIPE  29           /* Illegal seek */
#define ESRCH   3            /* No such process */
#define EROFS   30           /* Read-only file system */
#define EMLINK  31           /* Too many links */
#define EPIPE   32           /* Broken pipe */
#define EDOM    33           /* Math argument out of domain of
func */
#define ERANGE  34           /* Math result not representable */
#define EDEADLK 35           /* Resource deadlock would occur */
#define ENAMETOOLONG    36   /* File name too long */
#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 ELNRNG  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 EDOTDOT 73           /* RFS specific error */
#define EBADMSG 74           /* Not a data message */
#define EOVERFLOW       75   /* Value too large for defined data
type */
```



```
#define ENOTUNIQ       76     /* Name not unique on network */
#define EBADFD  77            /* File descriptor in bad state */
#define EREMCHG 78            /* Remote address changed */
#define ELIBACC 79              /* Can not access a needed shared
library */
#define ENOEXEC 8             /* Exec format error */
#define ELIBBAD 80                /* Accessing a corrupted shared
library */
#define ELIBSCN 81              /* .lib section in a.out corrupted
*/
#define ELIBMAX 82               /* Attempting to link in too many
shared libraries */
#define ELIBEXEC         83       /* Cannot exec a shared library
directly */
#define EILSEQ  84            /* Illegal byte sequence */
#define ERESTART       85     /* Interrupted system call should be
restarted */
#define ESTRPIPE       86     /* Streams pipe error */
#define EUSERS  87            /* Too many users */
#define ENOTSOCK       88     /* Socket operation on non-socket */
#define EDESTADDRREQ    89     /* Destination address required */
#define EBADF   9            /* Bad file number */
#define EMSGSIZE       90     /* Message too long */
#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);
```

14.4.12 error.h

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

14.4.13 execinfo.h

```
extern int backtrace(void **__array, int __size);
extern char **backtrace_symbols(void *const *__array, int __size);
extern void backtrace_symbols_fd(void *const *__array, int __size,
                                 int __fd);
```

14.4.14 fcntl.h

```
#define AT_FDCWD       -100    /* Use the current working directory
to determine the target of relative file paths. */
#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_RDWR  02
#define O_EXCL  0200
#define O_APPEND       02000
#define O_ASYNC 020000
#define O_CLOEXEC         02000000          /* The FD_CLOEXEC flag
associated with the new descriptor shall be set to close the file
descriptor upon execution of an exec family function. */
#define O_NOCTTY       0400
#define O_NDELAY       04000
#define O_NONBLOCK     04000
#define O_SYNC  04010000
#define AT_SYMLINK_NOFOLLOW     0x100   /* Do not follow symbolic
links. */
#define AT_EMPTY_PATH   0x1000  /* Allow empty relative pathname.
*/
#define AT_EACCESS     0x200   /* Check access using effective user
and group ID. */
#define AT_REMOVEDIR    0x200   /* Remove directory instead of file.
*/
#define AT_SYMLINK_FOLLOW      0x400   /* Follow symbolic link. */
#define  AT_NO_AUTOMOUNT  0x800     /*  Suppress  terminal  automount
traversal. */
#define FD_CLOEXEC     1
#define POSIX_FADV_RANDOM      1
#define F_DUPFD_CLOEXEC 1030    /* Duplicate file descriptor with
the close-on-exec flag FD_CLOEXEC set. */
#define POSIX_FADV_SEQUENTIAL   2
#define POSIX_FADV_WILLNEED    3
#define O_RSYNC O_SYNC

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
#ifndef SEEK_SET
#define SEEK_SET       0
#endif
#define F_GETFD 1
#define F_WRLCK 1
#ifndef SEEK_CUR
#define SEEK_CUR       1
#endif
#define F_SETSIG       10
#define F_GETSIG       11
#define F_SETFD 2
#define F_UNLCK 2
#ifndef SEEK_END
```

```
#define SEEK_END       2
#endif
#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);
```

14.4.15 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);
```

14.4.16 fnmatch.h

```
#define FNM_CASEFOLD    (1<<4)
#define FNM_FILE_NAME   FNM_PATHNAME

#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);
```

14.4.17 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);
```

14.4.18 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);
```

14.4.19 glob.h

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

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

typedef struct {
    size_t gl_pathc;
    char **gl_pathv;
    size_t gl_offs;
```

```
    int gl_flags;
    void (*gl_closedir) (void *);
    struct dirent *(*gl_readdir) (void *);
    void *(*gl_opendir) (const char *);
    int (*gl_lstat) (const char *, struct stat *);
    int (*gl_stat) (const char *, struct stat *);
} glob_t;

typedef struct {
    size_t gl_pathc;
    char **gl_pathv;
    size_t gl_offs;
    int gl_flags;
    void (*gl_closedir) (void *);
    struct dirent64 *(*gl_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);
```

14.4.20 gnu/libc-version.h

```
extern const char *gnu_get_libc_release(void);
extern const char *gnu_get_libc_version(void);
```

14.4.21 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 int setgroups(size_t __n, const gid_t * __groups);
```

14.4.22 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);
```

14.4.23 ifaddrs.h

```
#define ifa_broadaddr   ifa_ifu.ifu_broadaddr
#define ifa_dstaddr     ifa_ifu.ifu_dstaddr

struct ifaddrs {
    struct ifaddrs *ifa_next;
    char *ifa_name;
    unsigned int ifa_flags;
    struct sockaddr *ifa_addr;
    struct sockaddr *ifa_netmask;
    union {
        struct sockaddr *ifu_broadaddr;
        struct sockaddr *ifu_dstaddr;
    } ifa_ifu;
    void *ifa_data;
};
extern void freeifaddrs(struct ifaddrs *);
extern int getifaddrs(struct ifaddrs **);
```

14.4.24 inttypes.h

```
#if !defined __cplusplus || defined __STDC_FORMAT_MACROS
#define PRId16  "d"
#define PRId32  "d"
#define PRId8   "d"
#define PRIdFAST8       "d"
#define PRIdLEAST16     "d"
#define PRIdLEAST32     "d"
#define PRIdLEAST8      "d"
#define SCNd32  "d"
#define SCNdLEAST32     "d"
#define SCNd16  "hd"
#define SCNdLEAST16     "hd"
#define SCNd8   "hhd"
#define SCNdFAST8       "hhd"
#define SCNdLEAST8      "hhd"
#define SCNi8   "hhi"
#define SCNiFAST8       "hhi"
#define SCNiLEAST8      "hhi"
#define SCNo8   "hho"
#define SCNoFAST8       "hho"
#define SCNoLEAST8      "hho"
#define SCNu8   "hhu"
#define SCNuFAST8       "hhu"
#define SCNuLEAST8      "hhu"
#define SCNx8   "hhx"
#define SCNxFAST8       "hhx"
#define SCNxLEAST8      "hhx"
#define SCNi16  "hi"
#define SCNiLEAST16     "hi"
#define SCNo16  "ho"
#define SCNoLEAST16     "ho"
#define SCNu16  "hu"
#define SCNuLEAST16     "hu"
```

```
#define SCNx16  "hx"
#define SCNxLEAST16     "hx"
#define PRIi16  "i"
#define PRIi32  "i"
#define PRIi8   "i"
#define PRIiFAST8       "i"
#define PRIiLEAST16     "i"
#define PRIiLEAST32     "i"
#define PRIiLEAST8      "i"
#define SCNi32  "i"
#define SCNiLEAST32     "i"
#define PRIo16  "o"
#define PRIo32  "o"
#define PRIo8   "o"
#define PRIoFAST8       "o"
#define PRIoLEAST16     "o"
#define PRIoLEAST32     "o"
#define PRIoLEAST8      "o"
#define SCNo32  "o"
#define SCNoLEAST32     "o"
#define PRIu16  "u"
#define PRIu32  "u"
#define PRIu8   "u"
#define PRIuFAST8       "u"
#define PRIuLEAST16     "u"
#define PRIuLEAST32     "u"
#define PRIuLEAST8      "u"
#define SCNu32  "u"
#define SCNuLEAST32     "u"
#define PRIX16  "X"
#define PRIX32  "X"
#define PRIX8   "X"
#define PRIXFAST8       "X"
#define PRIXLEAST16     "X"
#define PRIXLEAST32     "X"
#define PRIXLEAST8      "X"
#define PRIx16  "x"
#define PRIx32  "x"
#define PRIx8   "x"
#define PRIxFAST8       "x"
#define PRIxLEAST16     "x"
#define PRIxLEAST32     "x"
#define PRIxLEAST8      "x"
#define SCNx32  "x"
#define SCNxLEAST32     "x"
#define PRId64  __PRI64_PREFIX"d"
#define PRIdFAST64      __PRI64_PREFIX"d"
#define PRIdLEAST64     __PRI64_PREFIX"d"
#define PRIdMAX __PRI64_PREFIX"d"
#define SCNd64  __PRI64_PREFIX"d"
#define SCNdFAST64      __PRI64_PREFIX"d"
#define SCNdLEAST64     __PRI64_PREFIX"d"
#define SCNdMAX __PRI64_PREFIX"d"
#define PRIi64  __PRI64_PREFIX"i"
#define PRIiFAST64      __PRI64_PREFIX"i"
#define PRIiLEAST64     __PRI64_PREFIX"i"
#define PRIiMAX __PRI64_PREFIX"i"
#define SCNi64  __PRI64_PREFIX"i"
#define SCNiFAST64      __PRI64_PREFIX"i"
#define SCNiLEAST64     __PRI64_PREFIX"i"
#define SCNiMAX __PRI64_PREFIX"i"
#define PRIo64  __PRI64_PREFIX"o"
#define PRIoFAST64      __PRI64_PREFIX"o"
#define PRIoLEAST64     __PRI64_PREFIX"o"
#define PRIoMAX __PRI64_PREFIX"o"
#define SCNo64  __PRI64_PREFIX"o"
```



```
#define SCNoFAST64      __PRI64_PREFIX"o"
#define SCNoLEAST64     __PRI64_PREFIX"o"
#define SCNoMAX __PRI64_PREFIX"o"
#define PRIu64  __PRI64_PREFIX"u"
#define PRIuFAST64      __PRI64_PREFIX"u"
#define PRIuLEAST64     __PRI64_PREFIX"u"
#define PRIuMAX __PRI64_PREFIX"u"
#define SCNu64  __PRI64_PREFIX"u"
#define SCNuFAST64      __PRI64_PREFIX"u"
#define SCNuLEAST64     __PRI64_PREFIX"u"
#define SCNuMAX __PRI64_PREFIX"u"
#define PRIX64  __PRI64_PREFIX"X"
#define PRIXFAST64      __PRI64_PREFIX"X"
#define PRIXLEAST64     __PRI64_PREFIX"X"
#define PRIXMAX __PRI64_PREFIX"X"
#define PRIx64  __PRI64_PREFIX"x"
#define PRIxFAST64      __PRI64_PREFIX"x"
#define PRIxLEAST64     __PRI64_PREFIX"x"
#define PRIxMAX __PRI64_PREFIX"x"
#define SCNx64  __PRI64_PREFIX"x"
#define SCNxFAST64      __PRI64_PREFIX"x"
#define SCNxLEAST64     __PRI64_PREFIX"x"
#define SCNxMAX __PRI64_PREFIX"x"
#define PRIdFAST16      __PRIPTR_PREFIX"d"
#define PRIdFAST32      __PRIPTR_PREFIX"d"
#define PRIdPTR __PRIPTR_PREFIX"d"
#define SCNdFAST16      __PRIPTR_PREFIX"d"
#define SCNdFAST32      __PRIPTR_PREFIX"d"
#define SCNdPTR __PRIPTR_PREFIX"d"
#define PRIiFAST16      __PRIPTR_PREFIX"i"
#define PRIiFAST32      __PRIPTR_PREFIX"i"
#define PRIiPTR __PRIPTR_PREFIX"i"
#define SCNiFAST16      __PRIPTR_PREFIX"i"
#define SCNiFAST32      __PRIPTR_PREFIX"i"
#define SCNiPTR __PRIPTR_PREFIX"i"
#define PRIoFAST16      __PRIPTR_PREFIX"o"
#define PRIoFAST32      __PRIPTR_PREFIX"o"
#define PRIoPTR __PRIPTR_PREFIX"o"
#define SCNoFAST16      __PRIPTR_PREFIX"o"
#define SCNoFAST32      __PRIPTR_PREFIX"o"
#define SCNoPTR __PRIPTR_PREFIX"o"
#define PRIuFAST16      __PRIPTR_PREFIX"u"
#define PRIuFAST32      __PRIPTR_PREFIX"u"
#define PRIuPTR __PRIPTR_PREFIX"u"
#define SCNuFAST16      __PRIPTR_PREFIX"u"
#define SCNuFAST32      __PRIPTR_PREFIX"u"
#define SCNuPTR __PRIPTR_PREFIX"u"
#define PRIXFAST16      __PRIPTR_PREFIX"X"
#define PRIXFAST32      __PRIPTR_PREFIX"X"
#define PRIXPTR __PRIPTR_PREFIX"X"
#define PRIxFAST16      __PRIPTR_PREFIX"x"
#define PRIxFAST32      __PRIPTR_PREFIX"x"
#define PRIxPTR __PRIPTR_PREFIX"x"
#define SCNxFAST16      __PRIPTR_PREFIX"x"
#define SCNxFAST32      __PRIPTR_PREFIX"x"
#define SCNxPTR __PRIPTR_PREFIX"x"
#endif

#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);
```

14.4.25 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);
```

14.4.26 libgen.h

```
#define basename __xpg_basename

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

14.4.27 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);
```

14.4.28 limits.h

```
#define LLONG_MIN      (-LLONG_MAX-1LL)
#define _POSIX_AIO_MAX  1
#define _POSIX_QLIMIT   1
#define _POSIX2_BC_STRING_MAX   1000
#define IOV_MAX 1024
#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 NAME_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_RE_DUP_MAX      255
#define _POSIX_SYMLINK_MAX     255
#define _POSIX_PATH_MAX 256
#define _POSIX_SEM_NSEMS_MAX    256
#define NGROUPS_MAX    32
#define WORD_BIT       32
#define _POSIX2_EXPR_NEST_MAX   32
#define _POSIX_DELAYTIMER_MAX   32
#define _POSIX_MQ_PRIO_MAX     32
#define _POSIX_SIGQUEUE_MAX    32
#define _POSIX_TIMER_MAX       32
#define _POSIX_SEM_VALUE_MAX    32767
#define _POSIX_SSIZE_MAX       32767
#define PATH_MAX       4096
#define _POSIX_ARG_MAX  4096
#define _POSIX_PIPE_BUF 512
#define _POSIX_TZNAME_MAX      6
#define _POSIX_LINK_MAX 8
#define _POSIX_MQ_OPEN_MAX     8
#define _POSIX_NGROUPS_MAX     8
#define _POSIX_RTSIG_MAX       8
#define _POSIX_STREAM_MAX      8
#define _POSIX_SYMLOOP_MAX     8
#define _POSIX_LOGIN_NAME_MAX   9
#define _POSIX_TTY_NAME_MAX    9
#define LLONG_MAX      9223372036854775807LL
#define _POSIX2_BC_BASE_MAX    99
#define _POSIX2_BC_SCALE_MAX   99
#define NL_MSGMAX      INT_MAX
#define NL_SETMAX      INT_MAX
#define NL_TEXTMAX     INT_MAX
#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 NL_LANGMAX     _POSIX2_LINE_MAX
#define NL_ARGMAX      _POSIX_ARG_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
```

14.4.29 link.h

```
extern int
dl_iterate_phdr(int  (*callback)  (struct  dl_phdr_info  *,  size_t,
void *),
             void *data);
```

14.4.30 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 *__ctype_tolower;
    const int *__ctype_toupper;
    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);
```

14.4.31 lsb/time.h

```
struct timeval {
    time_t tv_sec;
    suseconds_t tv_usec;
};
```

14.4.32 lsb/types.h

```
/*
 * This header is architecture dependent
 * Please refer to the specific architecture specification for
details
 */
```

14.4.33 lsb/wchar.h

```
typedef unsigned int wint_t;
typedef struct {
    int count;
    wint_t value;
} __mbstate_t;

typedef __mbstate_t mbstate_t;
```

14.4.34 monetary.h


```
extern ssize_t strfmon(char *__s, size_t __maxsize, const char
*__format,
                       ...);
extern ssize_t strfmon_l(char *s, size_t maxsize, locale_t locale,
                         const char *format, ...);
```

14.4.35 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    0x08    /* Is a loopback net. */
#define IFF_POINTOPOINT 0x10    /* Interface is point-to-point link.
*/
#define IFF_PROMISC    0x100   /* Receive all packets. */
#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       /* for use by
interface */
#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
*/
#define ifr_ifindex     ifr_ifru.ifru_ivalue    /* interface index
*/
#define ifr_metric     ifr_ifru.ifru_ivalue    /* metric */
#define ifr_qlen       ifr_ifru.ifru_ivalue    /* queue length */
#define ifr_mtu ifr_ifru.ifru_mtu      /* mtu */
#define ifr_netmask     ifr_ifru.ifru_netmask    /* interface net
mask */
#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_dstaddr;
        struct sockaddr ifru_broadaddr;
        struct sockaddr ifru_netmask;
        struct sockaddr ifru_hwaddr;
        short ifru_flags;
        int ifru_ivalue;
        int ifru_mtu;
        char ifru_slave[IFNAMSIZ];
        char ifru_newname[IFNAMSIZ];
        caddr_t ifru_data;
        struct ifmap ifru_map;
    } ifr_ifru;
};

#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);
```

14.4.36 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_errnop);
extern int gethostbyname_r(const char *__name,
                           struct hostent *__result_buf, char *__buf,
                           size_t __buflen, struct hostent **__result,
                           int *__h_errnop);
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);
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 struct servent *getservbyname(const char *__name,
                                     const char *__proto);
extern int getservbyname_r(const char *__name, const char *__proto,
                           struct servent *__result_buf, char *__buf,
                           size_t __buflen, struct servent **__result);
extern  struct  servent  *getservbyport(int  __port,  const  char
*__proto);
extern int getservbyport_r(int __port, const char *__proto,
                           struct servent *__result_buf, char *__buf,
                           size_t __buflen, struct servent **__result);
extern struct servent *getservent(void);
extern int getservent_r(struct servent *__result_buf, char *__buf,
                        size_t __buflen, struct servent **__result);
extern void setprotoent(int __stay_open);
extern void setservent(int __stay_open);
```

14.4.37 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_DST_UNREACH_ADMIN 1
#define ICMP6_FILTER    1
#define ICMP6_FILTER_BLOCK      1
#define ICMP6_PARAMPROB_NEXTHEADER      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 ND_OPT_PREFIX_INFORMATION       3
#define RPM_PCO_SETGLOBAL       3
#define ICMP6_DST_UNREACH_NOPORT        4
#define ICMP6_FILTER_PASSONLY   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_curhoplimit       nd_ra_hdr.icmp6_data8[0]
#define nd_ra_flags_reserved    nd_ra_hdr.icmp6_data8[1]
#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 {
    uint8_t nd_opt_type;
    uint8_t nd_opt_len;
};
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;
    uint16_t nd_opt_adv_interval_reserved;
    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;
};
```

14.4.38 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;
};
```

14.4.39 netinet/in.h

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

typedef uint16_t in_port_t;

struct in_addr {
    uint32_t s_addr;
};
typedef uint32_t in_addr_t;

#define INADDR_NONE     ((in_addr_t) 0xffffffff)
#define INADDR_BROADCAST       (0xffffffff)
#define INADDR_ANY      0
#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]
== 0 && ((const uint32_t *) (a))[1] == 0 && ((const uint32_t *)
(a))[2] == 0 && ((const uint32_t *) (a))[3] == 0)
#define IN6_IS_ADDR_LOOPBACK(a) (((const uint32_t *) (a))[0] == 0
&& ((const uint32_t *) (a))[1] == 0 && ((const uint32_t *) (a))[2]
== 0 && ((const uint32_t *) (a))[3] == htonl (1))
#define IN6_IS_ADDR_MULTICAST(a)       (((const uint8_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;
```

14.4.40 netinet/in_systm.h

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

14.4.41 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_RESTR       0xaf13
#define IPOPT_SECUR_MMMM        0xbc4d
#define IPOPT_SECUR_SECRET      0xd788
#define IPOPT_SECUR_CONFID      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 IPVERSION       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_LOWCOST   0x02
#define IPTOS_RELIABILITY       0x04
#define IPTOS_THROUGHPUT        0x08
#define IPTOS_LOWDELAY  0x10
#define IPTOS_TOS_MASK  0x1e
#define IPTOS_MINCOST   IPTOS_LOWCOST

#define IPTOS_PREC(tos) ((tos) & IPTOS_PREC_MASK)
#define IPTOS_PREC_MASK 0xe0
```

14.4.42 netinet/ip6.h

```
#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_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;
    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];
};
```

14.4.43 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_ANO   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 ICMP_PARAMPROB  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_IREQ       15
#define ICMP_PREC_CUTOFF        15
#define ICMP_UNREACH_PRECEDENCE_CUTOFF  15
#define NR_ICMP_UNREACH 15
#define ICMP_INFO_REPLY 16
#define ICMP_IREQREPLY  16
#define ICMP_ADDRESS    17
#define ICMP_MASKREQ    17
#define ICMP_ADDRESSREPLY       18
#define ICMP_MASKREPLY  18
#define ICMP_MAXTYPE    18
#define NR_ICMP_TYPES   18
#define ICMP_PROT_UNREACH       2
#define ICMP_REDIRECT_TOSNET    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_REDIR_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_SR_FAILED  5
#define ICMP_UNREACH_SRCFAIL    5
#define ICMP_NET_UNKNOWN       6
#define ICMP_UNREACH_NET_UNKNOWN       6
#define ICMP_HOST_UNKNOWN      7
#define ICMP_UNREACH_HOST_UNKNOWN      7
#define ICMP_ECHO      8
#define ICMP_HOST_ISOLATED     8
#define ICMP_MINLEN     8
#define ICMP_UNREACH_ISOLATED   8
#define ICMP_NET_ANO    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_hun.ih_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_addrs  icmp_hun.ih_rtradv.irt_num_addrs
#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 {
    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;
};
```

14.4.44 netinet/tcp.h

```
#define TCPOLEN_TSTAMP_APPA     (TCPOLEN_TIMESTAMP+2)
#define                                          TCPOPT_TSTAMP_HDR
(TCPOPT_NOP<<24|TCPOPT_NOP<<16|TCPOPT_TIMESTAMP<<8|TCPOLEN_TIMEST
AMP)
#define TCPOPT_EOL      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_SYNCNT      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_rtt;
    uint32_t tcpi_rttvar;
    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,
    TCP_CLOSING = 11
};
```

14.4.45 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;
};
```

14.4.46 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);
```

14.4.47 poll.h

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

14.4.48 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);
```

14.4.49 regex.h

```
#define RE_DUP_MAX      (0x7fff)

typedef unsigned long int reg_syntax_t;

typedef struct re_pattern_buffer {
    unsigned char *buffer;
    unsigned long int allocated;
    unsigned long int used;
```

```
    reg_syntax_t syntax;
    char *fastmap;
    char *translate;
    size_t re_nsub;
    unsigned int can_be_null:1;
    unsigned int regs_allocated:2;
    unsigned int fastmap_accurate:1;
    unsigned int no_sub:1;
    unsigned int not_bol:1;
    unsigned int not_eol:1;
    unsigned int newline_anchor:1;
} regex_t;
typedef int regoff_t;
typedef struct {
    regoff_t rm_so;
    regoff_t rm_eo;
} regmatch_t;

#define REG_ICASE      (REG_EXTENDED<<1)
#define REG_NEWLINE    (REG_ICASE<<1)
#define REG_NOSUB      (REG_NEWLINE<<1)
#define REG_EXTENDED   1

#define REG_NOTEOL     (1<<1)
#define REG_NOTBOL     1

typedef enum {
    REG_ENOSYS = -1,
    REG_NOERROR = 0,
    REG_NOMATCH = 1,
    REG_BADPAT = 2,
    REG_ECOLLATE = 3,
    REG_ECTYPE = 4,
    REG_EESCAPE = 5,
    REG_ESUBREG = 6,
    REG_EBRACK = 7,
    REG_EPAREN = 8,
    REG_EBRACE = 9,
    REG_BADBR = 10,
    REG_ERANGE = 11,
    REG_ESPACE = 12,
    REG_BADRPT = 13,
    REG_EEND = 14,
    REG_ESIZE = 15,
    REG_ERPAREN = 16
} 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);
```

14.4.50 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  (*ah_marshal)  (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 *);
```

14.4.51 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 CLSET_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_CANTENCODEARGS = 1,     /* can't encode arguments */
    RPC_CANTDECODERES = 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_PROGUNAVAIL = 8,        /* program not available */
    RPC_PROGVERSMISMATCH = 9,   /* program version mismatched */
    RPC_PROCUNAVAIL = 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_UNKNOWNPROTO = 17,      /* unknown protocol */
    RPC_UNKNOWNADDR = 19,       /* Remote address unknown */
    RPC_RPCBFAILURE = 14,       /* portmapper failed in its call */
    RPC_PROGNOTREGISTERED = 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 __recvsz);
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);
```

14.4.52 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);
```

14.4.53 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;
            xdrproc_t proc;
        } AR_results;
    } ru;
};

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

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 rp_acpt ru.RP_ar
#define rp_rjct ru.RP_dr

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);
```

14.4.54 rpc/svc.h

```
#define svc_getcaller(x)        (&(x)->xp_raddr)
#define svc_destroy(xprt)       (*(xprt)->xp_ops->xp_destroy)(xprt)
#define svc_recv(xprt,msg)       (*(xprt)->xp_ops->xp_recv)((xprt),
(msg))
#define svc_reply(xprt,msg)     (*(xprt)->xp_ops->xp_reply) ((xprt),
(msg))
#define svc_stat(xprt)  (*(xprt)->xp_ops->xp_stat)(xprt)
#define RPC_ANYSOCK     -1
#define svc_freeargs(xprt,xargs,argsp) \
       (*(xprt)->xp_ops->xp_freeargs)((xprt), (xargs), (argsp))
#define svc_getargs(xprt,xargs, argsp)  \
       (*(xprt)->xp_ops->xp_getargs)((xprt), (xargs), (argsp))

enum xprt_stat {
    XPRT_DIED,
    XPRT_MOREREQS,
    XPRT_IDLE
};

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

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

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

```
struct xp_ops {
   bool_t(*xp_recv) (SVCXPRT * __xprt, struct rpc_msg * __msg);
   enum xprt_stat (*xp_stat) (SVCXPRT * __xprt);
    bool_t(*xp_getargs) (SVCXPRT * __xprt, xdrproc_t __xdr_args,
                         caddr_t args_ptr);
    bool_t(*xp_reply) (SVCXPRT * __xprt, struct rpc_msg * __msg);
    bool_t(*xp_freeargs) (SVCXPRT * __xprt, xdrproc_t __xdr_args,
                          caddr_t args_ptr);
   void (*xp_destroy) (SVCXPRT * __xprt);
};
extern void svc_getreqset(fd_set * __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);
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 *svcfd_create(int, unsigned int, unsigned int);
extern SVCXPRT *svcraw_create(void);
extern SVCXPRT *svctcp_create(int __sock, u_int __sendsize,
                              u_int __recvsize);
extern SVCXPRT *svcudp_create(int __sock);
```

14.4.55 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;
```

14.4.56 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,
pos)
#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_handy;
} 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);
```

14.4.57 sched.h

```
#define __CPU_ALLOC_SIZE(count) ((((count) + __NCPUBITS - 1) /
__NCPUBITS) * 8)
#define __CPUELT(cpu)   ((cpu) / __NCPUBITS)
#define __CPUMASK(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_OP_S(setsize, destset, srcset1, srcset2, op)      \
  (__extension__\
   ({ cpu_set_t *__dest = (destset); \
     cpu_set_t *__arr1 = (srcset1); \
     cpu_set_t *__arr2 = (srcset2); \
     size_t __imax = (setsize) / sizeof (__cpu_mask); \
     size_t __i; \
     for (__i = 0; __i < __imax; ++__i)\
       __dest->__bits[__i]   =   __arr1->__bits[__i]   op   __arr2-
>__bits[__i]; \
  __dest; }))
#define __CPU_SET_S(cpu, setsize, cpusetp)      \
  (__extension__\
   ({ size_t __cpu = (cpu); \
```



```
    __cpu < 8 * (setsize) \
    ? ((cpusetp)->__bits[__CPUELT (__cpu)] |= __CPUMASK (__cpu)) :
0; }))
#define __CPU_ISSET_S(cpu, setsize, cpusetp)    \
  (__extension__\
   ({ size_t __cpu = (cpu); \
     __cpu < 8 * (setsize)\
    ?    (((cpusetp)->__bits[__CPUELT    (__cpu)]    &    __CPUMASK
(__cpu))) != 0 \
 : 0; }))
#define __CPU_CLR_S(cpu, setsize, cpusetp)      \
  (__extension__\
   ({ size_t __cpu = (cpu); \
     __cpu < 8 * (setsize)\
    ? ((cpusetp)->__bits[__CPUELT (__cpu)] &= ~__CPUMASK (__cpu)) :
0; }))
#define __CPU_ZERO_S(setsize, cpusetp)  \
  do {\
    size_t __i; \
    size_t __imax = (setsize) / sizeof (__cpu_mask); \
    cpu_set_t *__arr = (cpusetp); \
    for (__i = 0; __i < __imax; ++__i)\
      __arr->__bits[__i] = 0; \
  } while (0)
#define CPU_ALLOC_SIZE(count)   __CPU_ALLOC_SIZE (count)
#define   CPU_CLR(cpu,   cpusetp)        __CPU_CLR_S   (cpu,   sizeof
(cpu_set_t), cpusetp)
#define   CPU_ISSET(cpu,   cpusetp)   __CPU_ISSET_S   (cpu,   sizeof
(cpu_set_t), cpusetp)
#define CPU_AND_S(setsize, destset, srcset1, srcset2)   __CPU_OP_S
(setsize, destset, srcset1, srcset2, &)
#define CPU_XOR_S(setsize, destset, srcset1, srcset2)   __CPU_OP_S
(setsize, destset, srcset1, srcset2, ^)
#define CPU_OR_S(setsize, destset, srcset1, srcset2)    __CPU_OP_S
(setsize, destset, srcset1, srcset2, |)
#define CPU_AND(destset, srcset1, srcset2)      __CPU_OP_S (sizeof
(cpu_set_t), destset, srcset1, srcset2, &)
#define CPU_XOR(destset, srcset1, srcset2)      __CPU_OP_S (sizeof
(cpu_set_t), destset, srcset1, srcset2, ^)
#define CPU_OR(destset, srcset1, srcset2)       __CPU_OP_S (sizeof
(cpu_set_t), destset, srcset1, srcset2, |)
#define CPU_SETSIZE     __CPU_SETSIZE
#define   CPU_SET(cpu,   cpusetp)        __CPU_SET_S   (cpu,   sizeof
(cpu_set_t), cpusetp)
#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_getscheduler(pid_t __pid);
extern int sched_rr_get_interval(pid_t __pid, struct timespec *__t);
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);
```

14.4.58 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 *const *__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);
```

14.4.59 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);
```

14.4.60 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 SIGIOT  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
#define si_fd   _sifields._sigpoll._fd
#define si_timer1       _sifields._timer._timer1
#define si_timer2       _sifields._timer._timer2
#define sigev_notify_attributes _sigev_un._sigev_thread._attribute
#define sigev_notify_function   _sigev_un._sigev_thread._function

typedef struct siginfo {
   int si_signo;             /* Signal number. */
```



```
    int si_errno;
    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.
*/
#define FPE_FLTOVF     4       /* Floating-point overflow. */
#define FPE_FLTUND     5       /*  Floating-point underflow. */
#define FPE_FLTRES      6       /*  Floating-point inexact result.
*/
#define FPE_FLTINV     7       /* Invalid floating-point operation.
*/
#define FPE_FLTSUB     8       /* Subscript out of range. */
```

```
#define SEGV_MAPERR     1       /* Address not mapped to object. */
#define SEGV_ACCERR      2       /*  Invalid permissions for mapped
object. */

#define BUS_ADRALN      1       /*  Invalid address alignment. */
#define BUS_ADRERR      2       /*  Nonexistent physical address. */
#define BUS_OBJERR      3       /*  Object-specific hardware error.
*/

#define TRAP_BRKPT      1       /*  Process breakpoint. */
#define TRAP_TRACE      2       /*  Process trace trap. */

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

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

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

#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 using
`sa_restorer`. */
#define SA_RESETHAND    0x80000000      /* Reset to SIG_DFL on entry
to handler. */
#define SA_NOCLDSTOP    0x00000001      /* Don't send SIGCHLD when
children stop. */
#define SA_SIGINFO      0x00000004      /* Invoke signal-catching
function with three arguments instead of one. */
#define SA_NODEFER      0x40000000      /* Don't automatically block
the signal when its handler is being executed. */
#define SA_RESTART      0x10000000      /* Restart syscall on signal
return. */
#define SA_NOCLDWAIT    0x00000002      /* Don't create zombie on
child death. */
#define SA_NOMASK       SA_NODEFER
#define SA_ONESHOT      SA_RESETHAND

typedef struct sigaltstack {
   void *ss_sp;
   int ss_flags;
   size_t ss_size;
} stack_t;

#define SS_ONSTACK      1
#define SS_DISABLE      2

extern int __libc_current_sigrtmax(void);
extern int __libc_current_sigrtmin(void);
extern   sighandler_t   __sysv_signal(int   __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 __pgrp, int __sig);
extern void psiginfo(const siginfo_t * pinfo, const char *message);
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 raise(int __sig);
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 sigdelset(sigset_t * __set, int __signo);
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 __handler);
extern int sigorset(sigset_t * __set, const sigset_t * __left,
                const sigset_t * __right);
extern int sigpending(sigset_t * __set);
extern int sigprocmask(int __how, const sigset_t * __set,
                  sigset_t * __oset);
extern int sigqueue(pid_t __pid, int __sig, const union sigval
__val);
extern int sigrelse(int __sig);
extern int sigreturn(struct sigcontext *__scp);
extern sighandler_t sigset(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);
```

14.4.61 spawn.h

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

typedef struct {
    int __allocated;
    int __used;
    struct __spawn_action *__actions;
    int __pad[16];
} posix_spawn_file_actions_t;
typedef struct {
    short __flags;
    pid_t __pgrp;
    sigset_t __sd;
    sigset_t __ss;
    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_spawnattr_setflags(posix_spawnattr_t * _attr,
                                  short int __flags);
extern int posix_spawnattr_setpgroup(posix_spawnattr_t * __attr,
                                   pid_t __pgroup);
extern   int   posix_spawnattr_setschedparam(posix_spawnattr_t   *
__attr,
                                   const struct sched_param
                                   *__schedparam);
extern   int   posix_spawnattr_setschedpolicy(posix_spawnattr_t   *
__attr,
                                    int __schedpolicy);
extern   int   posix_spawnattr_setsigdefault(posix_spawnattr_t   *
__attr,
                                   const sigset_t * __sigdefault);
extern int posix_spawnattr_setsigmask(posix_spawnattr_t * __attr,
                                  const sigset_t * __sigmask);
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[]);
```


14.4.62 stddef.h

```
#if !defined(__GNUC__)
#define          __builtin_offsetof          (TYPE,          MEMBER)
((size_t)&((TYPE*)0)->MEMBER)
#endif
#ifndef NULL
#  ifdef __cplusplus
#    define NULL       (0L)
#  else
#    define NULL       ((void*) 0)
#  endif
#endif
#define offsetof(TYPE,MEMBER)   __builtin_offsetof (TYPE, MEMBER)
```

14.4.63 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       (-2147483647-1)
#define INT_LEAST32_MIN (-2147483647-1)
#define SIG_ATOMIC_MIN  (-2147483647-1)
#define INT16_MIN       (-32767-1)
#define INT_LEAST16_MIN (-32767-1)
#define INT64_MIN       (-__INT64_C(9223372036854775807)-1)
#define INTMAX_MIN      (-__INT64_C(9223372036854775807)-1)
#define INT_FAST64_MIN  (-__INT64_C(9223372036854775807)-1)
#define INT_LEAST64_MIN (-__INT64_C(9223372036854775807)-1)
#define WINT_MIN        (0u)
#define INT8_MAX        (127)
#define INT_FAST8_MAX   (127)
#define INT_LEAST8_MAX  (127)
#define INT32_MAX       (2147483647)
#define INT_LEAST32_MAX (2147483647)
#define SIG_ATOMIC_MAX  (2147483647)
#define UINT8_MAX       (255)
#define UINT_FAST8_MAX  (255)
#define UINT_LEAST8_MAX (255)
#define INT16_MAX       (32767)
#define INT_LEAST16_MAX (32767)
#define UINT32_MAX      (4294967295U)
#define UINT_LEAST32_MAX        (4294967295U)
#define WINT_MAX        (4294967295u)
#define UINT16_MAX      (65535)
#define UINT_LEAST16_MAX        (65535)
#define INT64_MAX       (__INT64_C(9223372036854775807))
#define INTMAX_MAX      (__INT64_C(9223372036854775807))
#define INT_FAST64_MAX  (__INT64_C(9223372036854775807))
#define INT_LEAST64_MAX (__INT64_C(9223372036854775807))
#define UINT64_MAX      (__UINT64_C(18446744073709551615))
#define UINTMAX_MAX     (__UINT64_C(18446744073709551615))
#define UINT_FAST64_MAX (__UINT64_C(18446744073709551615))
#define UINT_LEAST64_MAX        (__UINT64_C(18446744073709551615))

typedef signed char int8_t;
```

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

14.4.64 stdio.h

```
#define EOF    (-1)
#define P_tmpdir       "/tmp"
#ifndef SEEK_SET
#define SEEK_SET       0
#endif
#ifndef SEEK_CUR
#define SEEK_CUR       1
#endif
#define FOPEN_MAX      16
#ifndef SEEK_END
#define SEEK_END       2
#endif
#define L_tmpnam       20
#define TMP_MAX 238328
#define FILENAME_MAX    4096
#define BUFSIZ  8192
#define L_ctermid      9
#define L_cuserid      9

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

typedef struct _IO_FILE FILE;

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

extern char *__fgets_chk(char *, size_t, int, FILE *);
extern char *__fgets_unlocked_chk(char *, size_t, int, FILE *);
extern size_t __fpending(FILE *);
extern int __fprintf_chk(FILE *, int, const char *, ...);
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 __vfprintf_chk(FILE *, int, const char *, va_list);
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 ftell(FILE * __stream);
extern off_t ftello(FILE * __stream);
extern loff_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 * __s);
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 puts(const char *__s);
extern int putw(int __w, FILE * __stream);
extern int remove(const char *__filename);
extern int rename(const char *__old, const char *__new);
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);
```

14.4.65 stdlib.h

```
#define MB_CUR_MAX      (__ctype_get_mb_cur_max())
#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);
extern long long int __strtoll_internal(const char *, char **, int,
int);
extern unsigned long int __strtoul_internal(const char *, char **,
int,
                                            int);
extern unsigned long long int __strtoull_internal(const char *,
char **,
                                                  int, int);
extern  size_t  __wcstombs_chk(char  *,  const  wchar_t  *,  size_t,
size_t);
extern int __wctomb_chk(char *, wchar_t, size_t);
extern long int a64l(const char *__s);
extern void abort(void);
extern int abs(int __x);
extern int atexit(void (*__func) (void));
extern double atof(const char *__nptr);
extern int atoi(const char *__nptr);
extern long int atol(const char *__nptr);
extern long long int atoll(const char *__nptr);
extern void *bsearch(const void *__key, const void *__base, size_t
__nmemb,
                     size_t __size, __compar_fn_t __compar);
extern void *calloc(size_t __nmemb, size_t __size);
extern div_t div(int __numer, int __denom);
extern double drand48(void);
extern   int   drand48_r(struct   drand48_data   *__buffer,   double
*__result);
extern char *ecvt(double __value, int __ndigit, int *__decpt, int
*__sign);
extern char **environ;
extern double erand48(unsigned short __xsubi[3]);
extern int erand48_r(unsigned short __xsubi[3],
                     struct drand48_data *__buffer, double *__result);
extern void exit(int __status);
extern char *fcvt(double __value, int __ndigit, int *__decpt, int
*__sign);
```

```
extern void free(void *__ptr);
extern char *gcvt(double __value, int __ndigit, char *__buf);
extern char *getenv(const char *__name);
extern int getloadavg(double __loadavg[], int __nelem);
extern int getsubopt(char **__optionp, char *const *__tokens,
                     char **__valuep);
extern int grantpt(int __fd);
extern char *initstate(unsigned int __seed, char *__statebuf,
                       size_t __statelen);
extern int initstate_r(unsigned int __seed, char *__statebuf,
                       size_t __statelen, struct random_data *__buf);
extern long int jrand48(unsigned short __xsubi[3]);
extern int jrand48_r(unsigned short __xsubi[3],
                     struct drand48_data *__buffer, long int
*__result);
extern char *l64a(long int __n);
extern long int labs(long int __x);
extern void lcong48(unsigned short __param[7]);
extern int lcong48_r(unsigned short __param[7],
                     struct drand48_data *__buffer);
extern ldiv_t ldiv(long int __numer, long int __denom);
extern long long int llabs(long long int __x);
extern lldiv_t lldiv(long long int __numer, long long int __denom);
extern long int lrand48(void);
extern int lrand48_r(struct drand48_data *__buffer, long int
*__result);
extern void *malloc(size_t __size);
extern int mblen(const char *__s, size_t __n);
extern size_t mbstowcs(wchar_t * __pwcs, const char *__s, size_t
__n);
extern int mbtowc(wchar_t * __pwc, const char *__s, size_t __n);
extern char *mkdtemp(char *__template);
extern int mkstemp(char *__template);
extern int mkstemp64(char *__template);
extern char *mktemp(char *__template);
extern long int mrand48(void);
extern int mrand48_r(struct drand48_data *__buffer, long int
*__result);
extern long int nrand48(unsigned short __xsubi[3]);
extern int nrand48_r(unsigned short __xsubi[3],
                     struct drand48_data *__buffer, long int
*__result);
extern int posix_memalign(void **__memptr, size_t __alignment,
                          size_t __size);
extern int posix_openpt(int __oflag);
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 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 void srandom(unsigned int __seed);
extern int srandom_r(unsigned int __seed, 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 long long int strtoq(const char *__nptr, char **__endptr,
                            int __base);
extern unsigned long int strtoul(const char *__nptr, char
**__endptr,
                                 int __base);
extern unsigned long long int strtoull(const char *__nptr, char
**__endptr,
                                       int __base);
extern unsigned long long int strtouq(const char *__nptr, char
**__endptr,
                                      int __base);
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);
```

14.4.66 string.h

```
#define strerror_r __xpg_strerror_r

#define bzero(s,n)     memset(s,0,n)

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 *__stpncpy_chk(char *, const char *, size_t, 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 void *memchr(const void *__s, int __c, size_t __n);
extern int memcmp(const void *__s1, const void *__s2, size_t __n);
extern void *memcpy(void *__dest, const void *__src, size_t __n);
extern void *memmem(const void *__haystack, size_t __haystacklen,
                    const void *__needle, size_t __needlelen);
extern void *memmove(void *__dest, const void *__src, size_t __n);
extern void *memrchr(const void *__s, int __c, size_t __n);
extern void *memset(void *__s, 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 int strcoll_l(const char *s1, const char *s2, locale_t
locale);
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 char *strerror_l(int errnum, locale_t locale);
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 char *strrchr(const char *__s, int __c);
extern char *strsep(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);
extern size_t strxfrm_l(char *s1, const char *s2, size_t n,
                        locale_t locale);
```

14.4.67 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_l(const char *s1, const char *s2, locale_t
locale);
extern int strncasecmp(const char *__s1, const char *__s2, size_t
__n);
extern int strncasecmp_l(const char *s1, const char *s2, size_t n,
                         locale_t locale);
```

14.4.68 sys/epoll.h

```
#define EPOLL_CTL_ADD    1        /* Add a file decriptor to the
interface. */
#define EPOLL_CTL_DEL   2      /* Remove a file decriptor from the
interface. */
#define EPOLL_CTL_MOD   3      /* Change file decriptor epoll_event
structure. */
#define EPOLLIN 1
#define EPOLLPRI       2
#define EPOLLOUT       4
#define EPOLLERR       8
#define EPOLLHUP       16
#define EPOLLRDHUP     0x2000
#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);
```

14.4.69 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);
```

14.4.70 sys/inotify.h

```
#define IN_ACCESS       0x00000001
#define IN_MODIFY       0x00000002
#define IN_ATTRIB       0x00000004
#define IN_CLOSE_WRITE  0x00000008
#define IN_CLOSE_NOWRITE        0x00000010
#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
#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, int __wd);
```

14.4.71 sys/ioctl.h

```
#define _IOC(dir,type,nr,size)  (((dir) << _IOC_DIRSHIFT) | ((type)
<<  _IOC_TYPESHIFT)  |  ((nr)  <<  _IOC_NRSHIFT)  |  ((size)  <<
_IOC_SIZESHIFT))
#define _IOC_DIR(nr)    (((nr) >> _IOC_DIRSHIFT) & _IOC_DIRMASK)
#define _IOC_NR(nr)     (((nr) >> _IOC_NRSHIFT) & _IOC_NRMASK)
#define _IOC_SIZE(nr)   (((nr) >> _IOC_SIZESHIFT) & _IOC_SIZEMASK)
#define _IOC_TYPE(nr)   (((nr) >> _IOC_TYPESHIFT) & _IOC_TYPEMASK)
#define _IOC_DIRMASK    ((1 << _IOC_DIRBITS)-1)
#define _IOC_NRMASK     ((1 << _IOC_NRBITS)-1)
#define _IOC_SIZEMASK   ((1 << _IOC_SIZEBITS)-1)
#define _IOC_TYPEMASK   ((1 << _IOC_TYPEBITS)-1)
#define IOC_INOUT       ((_IOC_WRITE|_IOC_READ) << _IOC_DIRSHIFT)
#define _IOC_TYPECHECK(t)       (sizeof(t))
#define _IOC_TYPESHIFT  (_IOC_NRSHIFT+_IOC_NRBITS)
#define IOC_OUT (_IOC_READ << _IOC_DIRSHIFT)
#define IOCSIZE_MASK    (_IOC_SIZEMASK << _IOC_SIZESHIFT)
#define IOCSIZE_SHIFT   (_IOC_SIZESHIFT)
#define _IOC_DIRSHIFT   (_IOC_SIZESHIFT+_IOC_SIZEBITS)
#define _IOC_SIZESHIFT  (_IOC_TYPESHIFT+_IOC_TYPEBITS)
#define IOC_IN  (_IOC_WRITE << _IOC_DIRSHIFT)
#define _IOC_NRSHIFT    0
#define _IOC_NONE       0U
#define _IOC_SIZEBITS   14
#define _IOC_WRITE      1U
#define _IOC_DIRBITS    2
#define _IOC_READ       2U
#define _IOC_NRBITS     8
#define _IOC_TYPEBITS   8
#define _IO(type,nr)    _IOC(_IOC_NONE,(type),(nr),0)
#define                                                 _IOR(type,nr,size)
_IOC(_IOC_READ,(type),(nr),(_IOC_TYPECHECK(size)))
#define                                           _IOR_BAD(type,nr,size)
_IOC(_IOC_READ,(type),(nr),sizeof(size))
#define                                                _IOWR(type,nr,size)
_IOC(_IOC_READ|_IOC_WRITE,(type),(nr),(_IOC_TYPECHECK(size)))
#define                                          _IOWR_BAD(type,nr,size)
_IOC(_IOC_READ|_IOC_WRITE,(type),(nr),sizeof(size))
#define                                                 _IOW(type,nr,size)
_IOC(_IOC_WRITE,(type),(nr),(_IOC_TYPECHECK(size)))
#define                                           _IOW_BAD(type,nr,size)
_IOC(_IOC_WRITE,(type),(nr),sizeof(size))

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, ...);
```

14.4.72 sys/ipc.h

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

extern key_t ftok(const char *__pathname, int __proj_id);
```

14.4.73 sys/mman.h

```
#define MAP_FAILED      ((void*)-1)
#define POSIX_MADV_NORMAL       0
#define PROT_NONE       0x0
#define MAP_SHARED      0x01
#define MAP_PRIVATE     0x02
#define PROT_READ       0x1
#define MAP_FIXED       0x10
#define PROT_WRITE      0x2
#define MAP_ANONYMOUS   0x20
#define PROT_EXEC       0x4
#define MREMAP_MAYMOVE  1
#define MS_ASYNC        1
#define POSIX_MADV_RANDOM       1
#define MREMAP_FIXED    2
#define MS_INVALIDATE   2
#define POSIX_MADV_SEQUENTIAL   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);
```

14.4.74 sys/msg.h

```
#define MSG_NOERROR     010000

extern int msgctl(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 int __msgtyp, int __msgflg);
extern int msgsnd(int __msqid, const void *__msgp, size_t __msgsz,
                int __msgflg);
```

14.4.75 sys/param.h

```
#define NOFILE  256
#define MAXPATHLEN      4096
```

14.4.76 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 POLLERR 0x0008          /* Error condition */
#define POLLHUP 0x0010          /* Hung up */
#define POLLNVAL        0x0020  /* Invalid request: fd not open */
#define POLLRDNORM      0x0040  /* Normal data may be read */
#define POLLRDBAND      0x0080  /* Priority data may be read */
#define POLLWRNORM      0x0100  /* Writing now will not block */
#define POLLWRBAND      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;
```

14.4.77 sys/ptrace.h

```
enum __ptrace_setoptions {
    PTRACE_O_TRACESYSGOOD = 0x00000001,
    PTRACE_O_TRACEFORK = 0x00000002,
    PTRACE_O_TRACEVFORK = 0x00000004,
    PTRACE_O_TRACECLONE = 0x00000008,
    PTRACE_O_TRACEEXEC = 0x00000010,
    PTRACE_O_TRACEVFORKDONE = 0x00000020,
    PTRACE_O_TRACEEXIT = 0x00000040,
    PTRACE_O_MASK = 0x0000007f
};
enum __ptrace_eventcodes {
    PTRACE_EVENT_FORK = 1,
    PTRACE_EVENT_VFORK = 2,
    PTRACE_EVENT_CLONE = 3,
    PTRACE_EVENT_EXEC = 4,
    PTRACE_EVENT_VFORK_DONE = 5,
    PTRACE_EVENT_EXIT = 6
};
extern long int ptrace(enum __ptrace_request, ...);
```

14.4.78 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 RLIMIT_FSIZE    1
#define RLIMIT_LOCKS    10
#define RLIMIT_SIGPENDING       11
#define RLIMIT_MSGQUEUE 12
#define RLIMIT_NICE     13
#define RLIMIT_RTPRIO   14
#define RLIMIT_RTTIME   15
#define RLIM_NLIMITS    16
#define RLIMIT_DATA     2
#define RLIMIT_STACK    3
```



```
#define RLIMIT_CORE     4
#define RLIMIT_RSS      5
#define RLIMIT_NPROC    6
#define RLIMIT_NOFILE   7
#define RLIMIT_MEMLOCK  8
#define RLIMIT_AS       9

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

struct rlimit {
    rlim_t rlim_cur;          /* 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_oublock;        /* 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(__rlimit_resource_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);
```

14.4.79 sys/select.h

```
#define                  FD_ISSET(d,set)                  (((set)-
>fds_bits[((d)/(8*sizeof(long)))]&(1L<<((d)%(8*sizeof(long)))))!=
0)
#define          FD_CLR(d,set)                             ((set)-
>fds_bits[((d)/(8*sizeof(long)))]&=~(1L<<((d)%(8*sizeof(long)))))
#define          FD_SET(d,set)                             ((set)-
>fds_bits[((d)/(8*sizeof(long)))]|=(1L<<((d)%(8*sizeof(long)))))
#define NFDBITS (8 * sizeof (long))
#define FD_SETSIZE     1024
#define FD_ZERO(fdsetp) bzero(fdsetp, sizeof(*(fdsetp)))

typedef struct {
   unsigned long int fds_bits[FD_SETSIZE / NFDBITS];
} fd_set;
extern int pselect(int __nfds, fd_set * __readfds, fd_set *
__writefds,
               fd_set * __exceptfds, const struct timespec
*__timeout,
               const sigset_t *__sigmask);
extern int select(int __nfds, fd_set * __readfds, fd_set *
__writefds,
              fd_set * __exceptfds, struct timeval *__timeout);
```

14.4.80 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);
```

14.4.81 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);
```

14.4.82 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);
```

14.4.83 sys/socket.h

```
#define CMSG_FIRSTHDR(msg)      ((size_t) (msg)->msg_controllen >=
sizeof(struct  cmsghdr)  ?  (struct  cmsghdr  *)(msg)->msg_control  :
(struct cmsghdr *) NULL)
#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_NXTHDR(mhdr,cmsg)  \
       (((cmsg) == NULL) ? CMSG_FIRSTHDR(mhdr) : \
        (((u_char *)(cmsg) + CMSG_ALIGN((cmsg)->cmsg_len) \
                     + CMSG_ALIGN(sizeof(struct cmsghdr)) > \
          (u_char      *)((mhdr)->msg_control)      +      (mhdr)-
>msg_controllen) ? \
          (struct cmsghdr *)NULL : \
          (struct  cmsghdr  *)((u_char  *)(cmsg)  +  CMSG_ALIGN((cmsg)-
>cmsg_len))))

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

typedef unsigned short sa_family_t;
```

```
typedef unsigned int socklen_t;

struct sockaddr {
    sa_family_t sa_family;
    char sa_data[14];
};
struct sockaddr_storage {
    sa_family_t ss_family;
    __ss_aligntype __ss_align;
    char __ss_padding[(128 - (2 * sizeof(__ss_aligntype)))];
};

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

#define AF_UNSPEC       0
#define AF_UNIX 1
#define AF_INET6        10
#define AF_INET 2

#define PF_INET AF_INET
#define PF_INET6        AF_INET6
#define PF_UNIX AF_UNIX
#define 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 SIOCGIFDSTADDR  0x8917
#define SIOCGIFBRDADDR  0x8919
#define SIOCGIFNETMASK  0x891b
#define SIOCGIFMTU      0x8921
#define SIOCGIFHWADDR   0x8927

#define SHUT_RD 0
```



```
#define SHUT_WR 1
#define SHUT_RDWR      2

#define MSG_WAITALL     0x100
#define MSG_TRUNC       0x20
#define MSG_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
__addr_len);
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]);
```

14.4.84 sys/stat.h

```
#define S_ISBLK(m)      (((m)&S_IFMT)==S_IFBLK)
#define S_ISCHR(m)      (((m)&S_IFMT)==S_IFCHR)
#define S_ISDIR(m)      (((m)&S_IFMT)==S_IFDIR)
#define S_ISFIFO(m)     (((m)&S_IFMT)==S_IFIFO)
#define S_ISLNK(m)      (((m)&S_IFMT)==S_IFLNK)
#define S_ISREG(m)      (((m)&S_IFMT)==S_IFREG)
#define S_ISSOCK(m)     (((m)&S_IFMT)==S_IFSOCK)
```

```
#define UTIME_NOW      ((1l << 30) - 1l)
#define UTIME_OMIT     ((1l << 30) - 2l)
#define S_TYPEISMQ(buf) ((buf)->st_mode - (buf)->st_mode)
#define S_TYPEISSEM(buf)       ((buf)->st_mode - (buf)->st_mode)
#define S_TYPEISSHM(buf)       ((buf)->st_mode - (buf)->st_mode)
#define S_IRWXU (S_IREAD|S_IWRITE|S_IEXEC)
#define S_IROTH (S_IRGRP>>3)
#define S_IRGRP (S_IRUSR>>3)
#define S_IRWXO (S_IRWXG>>3)
#define S_IRWXG (S_IRWXU>>3)
#define S_IWOTH (S_IWGRP>>3)
#define S_IWGRP (S_IWUSR>>3)
#define S_IXOTH (S_IXGRP>>3)
#define S_IXGRP (S_IXUSR>>3)
#define S_ISVTX 01000
#define S_IXUSR 0x0040
#define S_IWUSR 0x0080
#define S_IRUSR 0x0100
#define S_ISGID 0x0400
#define S_ISUID 0x0800
#define S_IFIFO 0x1000
#define S_IFCHR 0x2000
#define S_IFDIR 0x4000
#define S_IFBLK 0x6000
#define S_IFREG 0x8000
#define S_IFLNK 0xa000
#define S_IFSOCK       0xc000
#define S_IFMT  0xf000
#define st_atime       st_atim.tv_sec
#define st_ctime       st_ctim.tv_sec
#define st_mtime       st_mtim.tv_sec
#define S_IREAD S_IRUSR
#define S_IWRITE       S_IWUSR
#define S_IEXEC S_IXUSR

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 fstat(int __fd, struct stat *__buf);
extern int fstat64(int __fd, struct stat64 *__buf);
extern int fstatat(int __fd, const char *__file, struct stat *__buf,
                   int __flag);
```



```
extern int fstatat64(int __fd, const char *__file, struct stat64
*__buf,
                 int __flag);
extern int futimens(int fd, const struct timespec times[2]);
extern int lstat(const char *__file, struct stat *__buf);
extern int lstat64(const char *__file, struct stat64 *__buf);
extern int mkdir(const char *__path, mode_t __mode);
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 int mknod(const char *__path, mode_t __mode, dev_t __dev);
extern int mknodat(int __fd, const char *__path, mode_t __mode,
               dev_t __dev);
extern int stat(const char *__file, struct stat *__buf);
extern int stat64(const char *__file, struct stat64 *__buf);
extern mode_t umask(mode_t __mask);
extern int utimensat(int fd, const char *path,
                 const struct timespec times[2], int flags);
```

14.4.85 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);
```

14.4.86 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);
```

14.4.87 sys/sysinfo.h

```
struct sysinfo {
   long int uptime;          /* Seconds since boot */
   unsigned long int loads[3]; /* 1, 5, and 15 minute load averages
*/
   unsigned long int totalram; /* Total usable main memory size */
   unsigned long int freeram;  /* Available memory size */
   unsigned long int sharedram;        /* Amount of shared memory
*/
   unsigned long int bufferram;       /* Memory used by buffers */
   unsigned long int totalswap;       /* Total swap space size */
   unsigned long int freeswap; /* Swap space still available */
   unsigned short procs;      /* Number of current processes */
   unsigned short pad;        /* Padding for m68k */
   unsigned long int totalhigh;       /* Total high memory size */
   unsigned long int freehigh; /* Available high memory size */
   unsigned int mem_unit;     /* Memory unit size in bytes */
   char _f[20 - 2 * sizeof(long) - sizeof(int)];       /* Padding
for libc5 */
};
extern int sysinfo(struct sysinfo *info);
```

14.4.88 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 itimerval {
   struct timeval it_interval;
   struct timeval it_value;
};
extern int adjtime(const struct timeval *__delta,
                struct timeval *__olddelta);
extern int futimes(int fd, const struct timeval tv[2]);
extern int getitimer(__itimer_which_t __which, struct itimerval
*__value);
extern int gettimeofday(struct timeval *__tv, struct timezone
*__tz);
extern int lutimes(const char *filename, const struct timeval
tv[2]);
extern int setitimer(__itimer_which_t __which,
                const struct itimerval *__new,
                struct itimerval *__old);
extern int utimes(const char *__file, const struct timeval *__tvp);
```

14.4.89 sys/timeb.h

```
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. */
};
extern int ftime(struct timeb *__timebuf);
```

14.4.90 sys/times.h

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

14.4.91 sys/types.h

```
#ifndef FALSE
#define FALSE   0
#endif
#ifndef TRUE
#define TRUE    1
#endif

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 long long int off64_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;
typedef unsigned long long int dev_t;
typedef unsigned int mode_t;
typedef unsigned long int nlink_t;
typedef char *caddr_t;

typedef long int clock_t;
typedef long int time_t;
```

14.4.92 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);
```

14.4.93 sys/un.h

```
struct sockaddr_un {
   sa_family_t sun_family;    /* AF_UNIX */
   char sun_path[108];
};
```

14.4.94 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);
```

14.4.95 sys/wait.h

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

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);
```

14.4.96 sysexits.h

```
#define EX_OK   0          /* successful termination  */
#define EX_USAGE      64     /* command line usage error  */
#define EX__BASE      64     /* base value for error messages  */
#define EX_DATAERR     65     /* data format error  */
#define EX_NOINPUT     66     /* cannot open input  */
#define EX_NOUSER      67     /* addressee unknown  */
#define EX_NOHOST      68     /* host name unknown  */
#define EX_UNAVAILABLE  69     /* service unavailable  */
#define EX_SOFTWARE    70     /* internal software error  */
#define EX_OSERR       71     /* system error (e.g., cannot fork)
*/
#define EX_OSFILE      72     /* critical OS file missing  */
#define EX_CANTCREAT    73     /* cannot create (user) output file
*/
#define EX_IOERR       74     /* input/output error  */
#define EX_TEMPFAIL     75     /* temp failure; user is invited to
retry  */
#define EX_PROTOCOL    76     /* remote error in protocol  */
#define EX_NOPERM      77     /* permission denied  */
#define EX_CONFIG      78     /* configuration error  */
```

14.4.97 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 */
#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 */

extern 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);
```

14.4.98 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
#define TGREAD  00040
#define TUEXEC  00100
#define TUWRITE 00200
#define TUREAD  00400
#define TSVTX   01000
#define TSGID   02000
#define TSUID   04000
#define TVERSLEN       2
#define TMAGLEN 6
#define TMAGIC  "ustar"
```

14.4.99 termios.h

```
#define TCIFLUSH       0
#define TCOOFF  0
#define TCSANOW 0
#define BS0     0000000
#define CR0     0000000
#define FF0     0000000
#define NL0     0000000
#define TAB0    0000000
#define VT0     0000000
#define OPOST   0000001
#define OCRNL   0000010
#define ONOCR   0000020
#define ONLRET  0000040
#define OFILL   0000100
#define OFDEL   0000200
#define NL1     0000400
#define TCOFLUSH       1
#define TCOON   1
#define TCSADRAIN      1
#define TCIOFF  2
#define TCIOFLUSH      2
#define TCSAFLUSH      2
#define TCION   3

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

#define NCCS    32

struct termios {
    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
#define PARMRK  0000010
#define INPCK   0000020
#define ISTRIP  0000040
#define INLCR   0000100
#define IGNCR   0000200
#define ICRNL   0000400
#define IXANY   0004000
#define IMAXBEL 0020000

#define CS5     0000000

#define ECHO    0000010

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

extern speed_t cfgetispeed(const struct termios *__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);
```

14.4.100 time.h

```
#define CLK_TCK ((clock_t)sysconf(2))
#define timerclear(tvp) ((tvp)->tv_sec = (tvp)->tv_usec = 0)
```

```
#define timerisset(tvp) ((tvp)->tv_sec || (tvp)->tv_usec)
#define CLOCK_REALTIME  0
#define CLOCK_MONOTONIC 1
#define TIMER_ABSTIME   1
#define CLOCKS_PER_SEC  10000001
#define CLOCK_PROCESS_CPUTIME_ID       2
#define CLOCK_THREAD_CPUTIME_ID 3
#define timeradd(a,b,result)    \
  do { \
    (result)->tv_sec = (a)->tv_sec + (b)->tv_sec; \
    (result)->tv_usec = (a)->tv_usec + (b)->tv_usec; \
    if ((result)->tv_usec >= 1000000) \
      { \
        ++(result)->tv_sec; \
        (result)->tv_usec -= 1000000; \
      } \
  } while (0)
#define timersub(a,b,result)    \
  do { \
    (result)->tv_sec = (a)->tv_sec - (b)->tv_sec; \
    (result)->tv_usec = (a)->tv_usec - (b)->tv_usec; \
    if ((result)->tv_usec < 0) { \
      --(result)->tv_sec; \
      (result)->tv_usec += 1000000; \
    } \
  } while (0)
#define timercmp(a,b,CMP)       \
 (((a)->tv_sec == (b)->tv_sec) ? \
  ((a)->tv_usec CMP (b)->tv_usec) : \
  ((a)->tv_sec CMP (b)->tv_sec))

struct tm {
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
    long int tm_gmtoff;
    char *tm_zone;
};

struct timespec {
    time_t tv_sec;
    long int tv_nsec;
};
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 size_t strftime_l(char *s, size_t maxsize, const char
*format,
                      const struct tm *timeptr, locale_t locale);
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);
```

14.4.101 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);
```

14.4.102 ulimit.h

```
#define UL_GETFSIZE    1
#define UL_SETFSIZE    2

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

14.4.103 unistd.h

```
#ifndef SEEK_SET
#define SEEK_SET       0
```

```
#endif
#define STDIN_FILENO    0
#ifndef SEEK_CUR
#define SEEK_CUR       1
#endif
#define STDOUT_FILENO   1
#ifndef SEEK_END
#define SEEK_END       2
#endif
#define STDERR_FILENO   2

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

#define _POSIX_VDISABLE '\0'
#define _POSIX_ASYNC_IO 0
#define _POSIX_CHOWN_RESTRICTED 1
#define _POSIX_JOB_CONTROL     1
#define _POSIX_NO_TRUNC 1
#define _POSIX_SHELL    1
#define _POSIX2_CHAR_TERM      200809L
#define _POSIX2_C_BIND  200809L
#define _POSIX2_LOCALEDEF      200809L
#define _POSIX2_VERSION 200809L
#define _POSIX_ADVISORY_INFO    200809L
#define _POSIX_BARRIERS 200809L
#define _POSIX_CLOCK_SELECTION  200809L
#define _POSIX_FSYNC    200809L
#define _POSIX_IPV6     200809L
#define _POSIX_MAPPED_FILES     200809L
#define _POSIX_MEMLOCK  200809L
#define _POSIX_MEMLOCK_RANGE    200809L
#define _POSIX_MEMORY_PROTECTION       200809L
#define _POSIX_MESSAGE_PASSING  200809L
#define _POSIX_PRIORITIZED_IO   200809L
#define _POSIX_PRIORITY_SCHEDULING     200809L
#define _POSIX_RAW_SOCKETS     200809L
#define _POSIX_READER_WRITER_LOCKS     200809L
#define _POSIX_REALTIME_SIGNALS 200809L
#define _POSIX_SEMAPHORES      200809L
#define _POSIX_SHARED_MEMORY_OBJECTS    200809L
#define _POSIX_SPAWN    200809L
#define _POSIX_SPIN_LOCKS      200809L
#define _POSIX_SYNCHRONIZED_IO  200809L
#define _POSIX_THREADS  200809L
#define _POSIX_THREAD_ATTR_STACKADDR    200809L
#define _POSIX_THREAD_ATTR_STACKSIZE    200809L
#define _POSIX_THREAD_PRIORITY_SCHEDULING      200809L
#define _POSIX_THREAD_PRIO_INHERIT     200809L
#define _POSIX_THREAD_PRIO_PROTECT     200809L
#define _POSIX_THREAD_PROCESS_SHARED    200809L
#define _POSIX_THREAD_ROBUST_PRIO_INHERIT      200809L
#define _POSIX_THREAD_SAFE_FUNCTIONS    200809L
#define _POSIX_TIMEOUTS 200809L
#define _POSIX_TIMERS   200809L
#define _POSIX_VERSION  200809L

#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_INT_MAX     104
#define _SC_INT_MIN     105
#define _SC_LONG_BIT    106
#define _SC_WORD_BIT    107
#define _SC_MB_LEN_MAX  108
#define _SC_NZERO       109
#define _SC_TIMERS      11
#define _SC_SSIZE_MAX   110
#define _SC_SCHAR_MAX   111
#define _SC_SCHAR_MIN   112
#define _SC_SHRT_MAX    113
#define _SC_SHRT_MIN    114
#define _SC_UCHAR_MAX   115
#define _SC_UINT_MAX    116
#define _SC_ULONG_MAX   117
#define _SC_USHRT_MAX   118
#define _SC_NL_ARGMAX   119
#define _SC_ASYNCHRONOUS_IO     12
#define _SC_NL_LANGMAX  120
#define _SC_NL_MSGMAX   121
#define _SC_NL_NMAX     122
#define _SC_NL_SETMAX   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_PRIORITIZED_IO      13
#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_C_LANG_SUPPORT      135
#define _SC_C_LANG_SUPPORT_R    136
#define _SC_CLOCK_SELECTION     137
#define _SC_CPUTIME     138
#define _SC_THREAD_CPUTIME      139
#define _SC_SYNCHRONIZED_IO     14
#define _SC_DEVICE_IO   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      15
#define _SC_MULTI_PROCESS      150
#define _SC_SINGLE_PROCESS     151
#define _SC_NETWORKING  152
#define _SC_READER_WRITER_LOCKS 153
#define _SC_SPIN_LOCKS  154
#define _SC_REGEXP     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_PBS_ACCOUNTING   169
#define _SC_MEMLOCK    17
#define _SC_2_PBS_LOCATE       170
#define _SC_2_PBS_MESSAGE      171
#define _SC_2_PBS_TRACK 172
#define _SC_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_TRACE_EVENT_FILTER  182
#define _SC_TRACE_INHERIT      183
#define _SC_TRACE_LOG  184
#define _SC_LEVEL1_ICACHE_SIZE  185
#define _SC_LEVEL1_ICACHE_ASSOC 186
#define _SC_LEVEL1_ICACHE_LINESIZE     187
#define _SC_LEVEL1_DCACHE_SIZE  188
#define _SC_LEVEL1_DCACHE_ASSOC 189
#define _SC_MEMORY_PROTECTION   19
#define _SC_LEVEL1_DCACHE_LINESIZE     190
#define _SC_LEVEL2_CACHE_SIZE   191
#define _SC_LEVEL2_CACHE_ASSOC  192
#define _SC_LEVEL2_CACHE_LINESIZE      193
#define _SC_LEVEL3_CACHE_SIZE   194
#define _SC_LEVEL3_CACHE_ASSOC  195
#define _SC_LEVEL3_CACHE_LINESIZE      196
#define _SC_LEVEL4_CACHE_SIZE   197
#define _SC_LEVEL4_CACHE_ASSOC  198
#define _SC_LEVEL4_CACHE_LINESIZE      199
#define _SC_CLK_TCK    2
#define _SC_MESSAGE_PASSING    20
#define _SC_SEMAPHORES  21
#define _SC_SHARED_MEMORY_OBJECTS      22
#define _SC_AIO_LISTIO_MAX     23
#define _SC_IPV6       235
#define _SC_RAW_SOCKETS 236
#define _SC_V7_ILP32_OFF32     237
```



```
#define _SC_V7_ILP32_OFFBIG     238
#define _SC_V7_LP64_OFF64       239
#define _SC_AIO_MAX     24
#define _SC_V7_LPBIG_OFFBIG     240
#define _SC_SS_REPL_MAX 241
#define _SC_TRACE_EVENT_NAME_MAX        242
#define _SC_TRACE_NAME_MAX      243
#define _SC_TRACE_SYS_MAX       244
#define _SC_TRACE_USER_EVENT_MAX        245
#define _SC_XOPEN_STREAMS       246
#define _SC_THREAD_ROBUST_PRIO_INHERIT  247
#define _SC_THREAD_ROBUST_PRIO_PROTECT  248
#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
#define _SC_PAGE_SIZE   30
#define _SC_RTSIG_MAX   31
#define _SC_SEM_NSEMS_MAX       32
#define _SC_SEM_VALUE_MAX       33
#define _SC_SIGQUEUE_MAX        34
#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_XOPEN_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_2_UPE      97
#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 _CS_POSIX_V6_ILP32_OFF32_CFLAGS 1116
#define _CS_POSIX_V6_ILP32_OFF32_LDFLAGS       1117
#define _CS_POSIX_V6_ILP32_OFF32_LIBS   1118
#define _CS_POSIX_V6_ILP32_OFF32_LINTFLAGS     1119
#define _CS_POSIX_V6_ILP32_OFFBIG_CFLAGS       1120
#define _CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS      1121
#define _CS_POSIX_V6_ILP32_OFFBIG_LIBS  1122
#define _CS_POSIX_V6_ILP32_OFFBIG_LINTFLAGS    1123
#define _CS_POSIX_V6_LP64_OFF64_CFLAGS  1124
#define _CS_POSIX_V6_LP64_OFF64_LDFLAGS 1125
#define _CS_POSIX_V6_LP64_OFF64_LIBS   1126
#define _CS_POSIX_V6_LP64_OFF64_LINTFLAGS      1127
#define _CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS       1128
#define _CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS      1129
#define _CS_POSIX_V6_LPBIG_OFFBIG_LIBS  1130
#define _CS_POSIX_V6_LPBIG_OFFBIG_LINTFLAGS    1131
#define _CS_POSIX_V7_ILP32_OFF32_CFLAGS 1132
```



```
#define _CS_POSIX_V7_ILP32_OFF32_LDFLAGS       1133
#define _CS_POSIX_V7_ILP32_OFF32_LIBS   1134
#define _CS_POSIX_V7_ILP32_OFF32_LINTFLAGS     1135
#define _CS_POSIX_V7_ILP32_OFFBIG_CFLAGS       1136
#define _CS_POSIX_V7_ILP32_OFFBIG_LDFLAGS      1137
#define _CS_POSIX_V7_ILP32_OFFBIG_LIBS  1138
#define _CS_POSIX_V7_ILP32_OFFBIG_LINTFLAGS    1139
#define _CS_POSIX_V7_LP64_OFF64_CFLAGS  1140
#define _CS_POSIX_V7_LP64_OFF64_LDFLAGS 1141
#define _CS_POSIX_V7_LP64_OFF64_LIBS    1142
#define _CS_POSIX_V7_LP64_OFF64_LINTFLAGS      1143
#define _CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS       1144
#define _CS_POSIX_V7_LPBIG_OFFBIG_LDFLAGS      1145
#define _CS_POSIX_V7_LPBIG_OFFBIG_LIBS  1146
#define _CS_POSIX_V7_LPBIG_OFFBIG_LINTFLAGS    1147
#define _CS_V6_ENV     1148
#define _CS_V7_ENV     1149

#define _XOPEN_XPG4    1
#define _XOPEN_VERSION  700

#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 *__getcwd_chk(char *, size_t, size_t);
extern int __getgroups_chk(int, 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 int __ttyname_r_chk(int, char *, size_t, size_t);
extern char **_environ;
extern void _exit(int __status);
extern int access(const char *__name, int __type);
extern int acct(const char *__name);
extern unsigned int alarm(unsigned int __seconds);
extern int brk(void *__addr);
extern int chdir(const char *__path);
extern int chown(const char *__file, uid_t __owner, gid_t __group);
extern int chroot(const char *__path);
extern int close(int __fd);
extern size_t confstr(int __name, char *__buf, size_t __len);
extern char *crypt(const char *__key, const char *__salt);
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 execl(const char *__path, const char *__arg, ...);
extern int execle(const char *__path, const char *__arg, ...);
extern int execlp(const char *__file, const char *__arg, ...);
extern int execv(const char *__path, char *const __argv[]);
extern int execve(const char *__path, char *const __argv[],
                  char *const __envp[]);
extern int execvp(const char *__file, char *const __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 char *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 gethostid(void);
extern int gethostname(char *__name, size_t __len);
extern char *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 char *getwd(char *__buf);
extern int isatty(int __fd);
extern int lchown(const char *__file, uid_t __owner, gid_t __group);
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 off_t lseek(int __fd, off_t __offset, int __whence);
extern loff_t lseek64(int __fd, loff_t __offset, int __whence);
extern int nice(int __inc);
extern char *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 rmdir(const char *__path);
extern void *sbrk(intptr_t __delta);
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 void 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 pid_t setsid(void);
extern int setuid(uid_t __uid);
extern unsigned int sleep(unsigned int __seconds);
extern void swab(const void *__from, void *__to, ssize_t __n);
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 long int sysconf(int __name);
extern pid_t tcgetpgrp(int __fd);
extern int tcsetpgrp(int __fd, pid_t __pgrp_id);
extern int truncate(const char *__file, off_t __length);
extern int truncate64(const char *__file, off64_t __length);
extern char *ttyname(int __fd);
extern int ttyname_r(int __fd, char *__buf, size_t __buflen);
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);
```

14.4.104 utime.h

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

14.4.105 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);
```

14.4.106 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);
```

14.4.107 wchar.h

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

typedef unsigned long int wctype_t;
typedef const int32_t *wctrans_t;

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 __mbsnrtowcs_chk(wchar_t *, const char **, size_t,
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 *__wcscpy_chk(wchar_t *, const wchar_t *, size_t);
extern wchar_t *__wcsncat_chk(wchar_t *, const wchar_t *, size_t,
size_t);
extern wchar_t *__wcsncpy_chk(wchar_t *, const wchar_t *, size_t,
size_t);
```



```
extern size_t __wcsnrtombs_chk(char *, const wchar_t * *, size_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);
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 fgetwc_unlocked(FILE * __stream);
extern wchar_t *fgetws(wchar_t * __ws, int __n, FILE * __stream);
extern wchar_t *fgetws_unlocked(wchar_t * __ws, int __n, FILE *
__stream);
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 int 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 mbrtowc(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 **__src,
size_t __nmc,
                         size_t __len, mbstate_t * __ps);
extern size_t mbsrtowcs(wchar_t * __dst, const char **__src, 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, ...);
extern wint_t ungetwc(wint_t __wc, FILE * __stream);
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 int vwprintf(const wchar_t * __format, va_list __arg);
extern int vwscanf(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 int wcscasecmp_l(const wchar_t * ws1, const wchar_t * ws2,
                        locale_t locale);
extern wchar_t *wcscat(wchar_t * __dest, const wchar_t * __src);
extern wchar_t *wcschr(const wchar_t * __wcs, wchar_t __wc);
extern int wcscmp(const wchar_t * __s1, const wchar_t * __s2);
extern int wcscoll(const wchar_t * __s1, const wchar_t * __s2);
extern int wcscoll_l(const wchar_t * ws1, const wchar_t * ws2,
                     locale_t locale);
extern wchar_t *wcscpy(wchar_t * __dest, const wchar_t * __src);
extern  size_t  wcscspn(const  wchar_t  *  __wcs,  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 wcsncasecmp(const wchar_t * __s1, const wchar_t * __s2,
                       size_t __n);
extern int wcsncasecmp_l(const wchar_t * ws1, const wchar_t * ws2,
                         size_t n, locale_t locale);
extern wchar_t *wcsncat(wchar_t * __dest, const wchar_t * __src,
                        size_t __n);
extern  int  wcsncmp(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);
extern size_t wcsnlen(const wchar_t * __s, size_t __maxlen);
extern  size_t  wcsnrtombs(char  *__dst,  const  wchar_t  *  *__src,
size_t __nwc,
                           size_t __len, mbstate_t * __ps);
extern  wchar_t  *wcspbrk(const  wchar_t  *  __wcs,  const  wchar_t  *
__accept);
extern wchar_t *wcsrchr(const wchar_t * __wcs, wchar_t __wc);
extern size_t wcsrtombs(char *__dst, const wchar_t * *__src, size_t
__len,
                        mbstate_t * __ps);
extern  size_t  wcsspn(const  wchar_t  *  __wcs,  const  wchar_t  *
__accept);
extern wchar_t *wcsstr(const wchar_t * __haystack,
                       const wchar_t * __needle);
extern double wcstod(const wchar_t * __nptr, wchar_t * *__endptr);
extern float wcstof(const wchar_t * __nptr, wchar_t * *__endptr);
extern wchar_t *wcstok(wchar_t * __s, const wchar_t * __delim,
                       wchar_t * *__ptr);
extern long int wcstol(const wchar_t * __nptr, wchar_t * *__endptr,
                       int __base);
extern  long  double  wcstold(const  wchar_t  *  __nptr,  wchar_t  *
*__endptr);
extern long long int wcstoll(const wchar_t * __nptr, wchar_t *
*__endptr,
                             int __base);
extern  long  long  int  wcstoq(const  wchar_t  *  __nptr,  wchar_t  *
*__endptr,
                            int __base);
extern unsigned long int wcstoul(const wchar_t * __nptr,
```

```
                                wchar_t * *__endptr, int __base);
extern unsigned long long int wcstoull(const wchar_t * __nptr,
                                       wchar_t * *__endptr, int __base);
extern unsigned long long int wcstouq(const wchar_t * __nptr,
                                      wchar_t * *__endptr, int __base);
extern wchar_t *wcswcs(const wchar_t * __haystack,
                       const wchar_t * __needle);
extern int wcswidth(const wchar_t * __s, size_t __n);
extern size_t wcsxfrm(wchar_t * __s1, const wchar_t * __s2, size_t
__n);
extern size_t wcsxfrm_l(const wchar_t * ws1, const wchar_t * ws2,
size_t n,
                        locale_t locale);
extern int wctob(wint_t __c);
extern int wcwidth(wchar_t __c);
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 *wmemmove(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, ...);
```

14.4.108 wctype.h

```
extern int iswalnum(wint_t __wc);
extern int iswalnum_l(wint_t wc, locale_t locale);
extern int iswalpha(wint_t __wc);
extern int iswalpha_l(wint_t wc, locale_t locale);
extern int iswblank(wint_t __wc);
extern int iswblank_l(wint_t wc, locale_t locale);
extern int iswcntrl(wint_t __wc);
extern int iswcntrl_l(wint_t wc, locale_t locale);
extern int iswctype(wint_t __wc, wctype_t __desc);
extern int iswctype_l(wint_t wc, locale_t locale);
extern int iswdigit(wint_t __wc);
extern int iswdigit_l(wint_t wc, locale_t locale);
extern int iswgraph(wint_t __wc);
extern int iswgraph_l(wint_t wc, locale_t locale);
extern int iswlower(wint_t __wc);
extern int iswlower_l(wint_t wc, locale_t locale);
extern int iswprint(wint_t __wc);
extern int iswprint_l(wint_t wc, locale_t locale);
extern int iswpunct(wint_t __wc);
extern int iswpunct_l(wint_t wc, locale_t locale);
extern int iswspace(wint_t __wc);
extern int iswspace_l(wint_t wc, locale_t locale);
extern int iswupper(wint_t __wc);
extern int iswupper_l(wint_t wc, locale_t locale);
extern int iswxdigit(wint_t __wc);
extern int iswxdigit_l(wint_t wc, locale_t locale);
extern wint_t towctrans(wint_t __wc, wctrans_t __desc);
extern wint_t towctrans_l(wint_t wc, wctrans_t desc, locale_t
locale);
extern wint_t towlower(wint_t __wc);
extern wint_t towlower_l(wint_t wc, locale_t locale);
extern wint_t towupper(wint_t __wc);
extern wint_t towupper_l(wint_t wc, locale_t locale);
extern wctrans_t wctrans(const char *__property);
extern size_t wctrans_l(const char *charclass, locale_t locale);
extern wctype_t wctype(const char *__property);
```

```
extern size_t wctype_l(const char *property, locale_t locale);
```

14.4.109 wordexp.h

```
enum {
    WRDE_DOOFFS = 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);
```

14.5 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 14.3 shall behave as described in the referenced base document.

_IO_feof

Name

`_IO_feof` — alias for feof

Synopsis

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

Description

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

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

_IO_getc

Name

_IO_getc — alias for getc

Synopsis

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

Description

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

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

_IO_putc

Name

_IO_putc — alias for putc

Synopsis

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

Description

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

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

_IO_puts

Name

_IO_puts — alias for puts

Synopsis

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

Description

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

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

__assert_fail

Name

__assert_fail — abort the program after false assertion

Synopsis

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

Description

The __assert_fail() function is used to implement the assert() interface of POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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

```
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.

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

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 current character set. The array shall contain a total of 384 characters, and can be indexed with any signed or unsigned char (i.e. with an index value between -128 and 255). If the application is multithreaded, the array shall be local to the current thread.

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

Return Value

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

__ctype_get_mb_cur_max

Name

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

Synopsis

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

Description

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

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

__ctype_tolower_loc

Name

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

Synopsis

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

Description

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

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

Return Value

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

__ctype_toupper_loc

Name

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

Synopsis

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

Description

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

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

Return Value

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

__cxa_atexit

Name

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

Synopsis

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

Description

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

The __cxa_atexit() function is used to implement atexit(), as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009). Calling atexit(func) from the statically linked part of an application shall be equivalent to __cxa_atexit(func, NULL, NULL).

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

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

__cxa_finalize

Name

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

Synopsis

```
void __cxa_finalize(void * d);
```

Description

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

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

The list is populated by entries of two kinds:

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

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

When __cxa_finalize(d) is called, it shall walk the termination function list, calling each in turn if d matches the handle of the termination function entry. If d is NULL, it shall call all the termination funtions. 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). __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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). __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

```
size_t __fpending(FILE * stream);
```

Description

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

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

__fprintf_chk

Name

__fprintf_chk — convert formatted output, with stack checking

Synopsis

```
#include <stdio.h>
int __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 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 __fprintf_chk() function is not in the source standard; it is only in the binary 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 (ISO/IEC 9945-2009).

__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, fstatat64

Name

__fxstatat64, fstatat64 — get file status relative to a directory file descriptor (Large File Support)

Synopsis

```
#include <fcntl.h>
#include <sys/stat.h>
int __fxstatat64(int ver, int dirfd, const char * path, struct stat64
* stat_buf, int flags);
int fstatat64(int dirfd, const char * file, struct stat64 * buf, int
flag);
```

Description

fstatat64() is a large-file version of the fstatat() function as defined in POSIX 1003.1-2008 (ISO/IEC 9945-2009). It differs from fstatat() only in that the *buf* parameter refers to a large-file version of the stat structure.

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*)

__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 interface getgroups(), except that __getgroups_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 *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 interface gethostname(), except that __gethostname_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 __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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), except that the argument type for __isnanf() is known to be float.

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

__isnanl

Name

__isnanl — test for infinity

Synopsis

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

Description

__isnanl() has the same specification as isnan() in POSIX 1003.1-2008 (ISO/IEC 9945-2009), except that the argument type for __isnanl() is known to be long double.

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

__libc_current_sigrtmax

Name

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

Synopsis

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

Description

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

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

__libc_current_sigrtmin

Name

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

Synopsis

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

Description

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

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

__libc_start_main

Name

__libc_start_main — initialization routine

Synopsis

```
int __libc_start_main(int (*main) (int, char **, char **), int argc,
char ** 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 the LSB Core Specification.

__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 binary 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 binary 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 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 __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 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 __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

```
#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

```
#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

```
#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

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

Description

__register_atfork() implements pthread_atfork() as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 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 *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

```
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.

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

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 binary standard.

__stpncpy_chk

Name

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

Synopsis

```
#include <string.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 binary 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 binary 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 strcpy(), 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 binary standard.

__strdup

Name

__strdup — alias for strdup

Synopsis

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

Description

__strdup() has the same specification as strdup().

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

__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 binary 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 binary standard.

__strtod_internal

Name

__strtod_internal — underlying function for strtod

Synopsis

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

Description

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

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

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

__strtof_internal

Name

__strtof_internal — underlying function for strtof

Synopsis

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

Description

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

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

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

__strtok_r

Name

__strtok_r — alias for strtok_r

Synopsis

```
char * __strtok_r(char * restrict s, const char * restrict delim,
char * * restrict save_ptr);
```

Description

__strtok_r() has the same specification as strtok_r().

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

__strtol_internal

Name

__strtol_internal — alias for strtol

Synopsis

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

Description

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

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

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

__strtold_internal

Name

__strtold_internal — underlying function for strtold

Synopsis

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

Description

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

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

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

__strtoll_internal

Name

__strtoll_internal — underlying function for strtoll

Synopsis

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

Description

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

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

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

__strtoul_internal

Name

__strtoul_internal — underlying function for strtoul

Synopsis

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

Description

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

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

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

__strtoull_internal

Name

__strtoull_internal — underlying function for strtoull

Synopsis

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

Description

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

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

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

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

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

__timezone

Name

__timezone — external variable containing timezone

Synopsis

```
long int __timezone;
```

Description

The external variable __timezone shall implement the timezone variable timezone as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009). __timezone has the same specification as timezone.

__ttyname_r_chk

Name

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

Synopsis

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

Description

The interface __ttyname_r_chk() shall function in the same way as the interface ttyname_r(), except that __ttyname_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 __ttyname_r_chk() function is not in the source standard; it is only in the binary standard.

__tzname

Name

__tzname — 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) function tzset(). __tzname has the same specification as tzname.

__vfprintf_chk

Name

__vfprintf_chk — convert formatted output, with stack checking

Synopsis

```
#include <stdio.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 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 __vfprintf_chk() function is not in the source standard; it is only in the binary standard.

__vfwprintf_chk

Name

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

Synopsis

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

Description

The interface __vfwprintf_chk() shall function in the same way as the interface vfwprintf(), except that __vfwprintf_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 __vfwprintf_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

```
#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

```
#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

```
#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 wcscpy(), 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

```
#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 binary 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 interface 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 binary 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

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 (ISO/IEC 9945-2009).

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) basename().

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

Return Value

See POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

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

Return Value

See POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 *strerrbuf*, with length *buflen*, as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009) strerror_r().

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

Return Value

See POSIX 1003.1-2008 (ISO/IEC 9945-2009).

__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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

`__lxstat(_STAT_VER, path, stat_buf)` shall implement `lstat(path, stat_buf)` as specified by POSIX 1003.1-2008 (ISO/IEC 9945-2009).

`__fxstat(_STAT_VER, fildes, stat_buf)` shall implement `fstat(fildes, stat_buf)` as specified by POSIX 1003.1-2008 (ISO/IEC 9945-2009).

`__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 signal description strings ordered by signal number.

The `_sys_siglist` array is only in the binary standard; it is not in the source standard. Applications wishing to access signal descriptions should use the `strsignal()` function.

acct

Name

`acct` — switch process accounting on or off

Synopsis

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

Description

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

Return Value

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

Errors

ENOSYS

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

ENOMEM

Out of memory.

EPERM

The calling process has no permission to enable process accounting.

EACCES

`filename` is not a regular file.

EIO

Error writing to the *`filename`*.

EUSERS

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

adjtime

Name

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

Synopsis

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

Description

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

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

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

`alpahsort64()` is a large-file version of the `alphasort()` function as defined in POSIX 1003.1-2008 (ISO/IEC 9945-2009). If differs only in that the *d1* and *d2* parameters are of type `dirent64` instead of type `dirent`.

argz_add, argz_add_sep, argz_append, argz_count, argz_create, argz_create_sep, argz_delete, argz_extract, argz_insert, argz_next, argz_replace, argz_stringify

Name

`argz_add, argz_add_sep, argz_append, argz_count, argz_create, argz_create_sep, argz_delete, argz_extract, argz_insert, argz_next, argz_replace, argz_stringify` — Operate on argz vectors

Synopsis

```
#include <argz.h>
error_t argz_add(char ** argz, size_t * argz_len, const char * str);
error_t argz_add_sep(char ** argz, size_t * argz_len, const char *
str, int sep);
error_t argz_append(char ** argz, size_t * argz_len, const char * buf,
size_t buf_len);
size_t argz_count(const char * argz, size_t * argz_len);
error_t argz_create(char * const argv, char ** argz, size_t * argz_len);
error_t argz_create_sep(const char * str, int sep, char ** argz, size_t
* argz_len);
void argz_delete(char ** argz, size_t * argz_len, char * entry);
void argz_extract(const char * argz, size_t argz_len, char ** argv);
error_t argz_insert(char ** argz_insert, size_t * argz_len, char *
before, const char * entry);
char argz_next(const char * argz, size_t argz_len, const char * entry);
error_t argz_replace(char ** argz, size_t * argz_len, const char *
str, const char * with, unsigned int * replace_count);
void argz_stringify(char * argz, size_t argz_len, int sep);
```

Description

The argz functions operate on argz vectors, which are typically used to more easily manipulate program arguments, of the form described in ISO C (1999) in section 5.1.2.2.1, Program Startup. While an argv is an array of character pointers to strings, an argz vector is a set of strings, separated by null characters, in contiguous memory, the vector is described by a pointer to the first element and a size. There is no limitation that the argz must be made up of program arguments.

The argz functions which change argz vectors expect them to use memory allocated using `malloc()`, and will themselves use `malloc()` or `realloc()`.

The `argz_create()` function converts an argv vector identified by `argv` to an argz vector with the same elements, identified by `argz` and `argz_len`.

The `argz_create_sep()` function converts the string identified by `str`, spliting into a separate string at each occurence of `sep`, to an argz vector identified by `argz` and `argz_len`.

The `argz_add()` function adds the string identified by `str` to the vector identified by `argz` and `argz_len`, updating `argz` and `argz_len`.

The `argz_add_sep()` function adds the string identified by `str`, spliting into a separate string at each occurence of `sep`, to the vector identified by `argz`, updating `argz` and `argz_len`.

The `argz_append()` function appends the argz vector identified by `buf` and `buf_len` to the argz vector identified by `argz` and `argz_len`, thus updating `argz` and `argz_len`.

The `argz_count()` function returns the number of strings in the argz vector identified by `argz` and `argz_len`.

The `argz_delete()` function removes the string identified by `entry` from the the argz vector identified by `argz`, `argz_len`, updating `argz` and `argz_len`.

The `argz_extract()` function performs the inverse of `argz_create()`. It converts an argz vector identified by `argz` and `argz_len` to an argv vector identified by `argv` with the same elements.

The `argz_insert()` function inserts the string identified by `entry` at position `before` to the the argz vector identified by `argz` and `argz_len`, updating `argz` and `argz_len`.

The `argz_next()` function returns the entry following the entry identfied by `entry` in the argz vector identified by `argz` and `argz_len`. If `entry` is `NULL` the first entry is returned. This function can be used to step through an argz vector by obtaining the first entry by passing `NULL`, then passing the just obtained value to the next call, and so on. `NULL` is returned if there is no following entry.

The `argz_replace()` function replaces each occurrence of `str` in the argz vector identified by `argz` and `argz_len` with `with`, updating `argz` and `argz_len`. The counter pointed to by `replace_count` will be incremented by the number of replacements unless `NULL` is passed for `replace_count`.

The `argz_stringify()` function performs the inverse of `argz_create_sep()`. It converts the argz vector identified by `argz` and `argz_len` into a regular string, with the strings in the original vector separated by `sep` in the converted string. The conversion is done in place, so in effect each null byte in `argz` but the last one is replaced by `sep`.

Return Value

All of the argz functions that perform memory allocation return an error_t type. These functions return `0` on success; if memory allocation fails, they return ENOMEM.

`argz_count()` returns a count of substrings in the argz vector as a size_t type.

`argz_next()` returns a pointer to a substring in an argz vector, or `NULL`.

See Also

envz_add, envz_entry, envz_get, envz_merge, envz_remove, envz_strip

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()`.

backtrace, backtrace_symbols, backtrace_symbols_fd

Name

`backtrace, backtrace_symbols, backtrace_symbols_fd` — runtime stack back tracing

Synopsis

```
#include <execinfo.h>
int backtrace(void **array, int size);
char **backtrace_symbols(void *const *array, int size);
void backtrace_symbols_fd(void *const *array, int size, int fd);
```

Description

`backtrace()` obtains a backtrace for the current thread as a list of pointers filled in to *`array`*. The *`size`* parameter describes the number of elements that will fit into *`array`*, `backtrace()` will truncate the list if necessary. A backtrace is a list of currently active function calls in a thread; each function call allocates a new stack frame and `backtrace()` obtains the return address from each stack frame.

`backtrace_symbols()` translates the information obtained from `backtrace()` into an array of strings. *`array`* is a pointer to an array of addresses as obtained from `backtrace()`. *`size`* is the number of entries in *`array`*, and should be the return value of the call to `backtrace()`. The strings contain the function name if it can be determined, a hedxadecimal offset into the function, and the actual return address in hexadecimal. Note that the pointer returned by `backtrace_symbols()` is obtained by an internal call to `malloc()` and should be freed when no longer needed.

`backtrace_symbols_fd()` performs the same transformation as `backtrace_symbols()` given the same argument pair *`array`*, *`size`*, but writes the strings to the file descriptor contained in *`fd`*. This avoids the allocation of string space.

Return Value

`backtrace()` returns the number of entries placed into *`array`*, no more than *`size`*. If the value is less than *`size`*, the full backtrace was returned; else it may have been truncated.

On success, `backtrace_symbols()` returns a pointer to an array of strings, which will have *`size`* entries. On error, `NULL` is returned.

Errors

No errors are defined for these functions. If `backtrace_symbols_fd()` fails, it will be due to a failure in the call to `malloc()`, and `errno` will be set accordingly.

Notes

The ability to obtain useful backtrace information, in particular function names, is dependent on a number of factors at the time of program construction, such as compiler optimization options. Even if the program itself is constructed so as to make symbols visible, the call trace may descend into system libraries which have not been so constructed.

Inlined functions do not have stack frames, and functions declared as static are not exposed and so will not be available in the backtrace.

See Also

`malloc()`

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), and is equivalent to the function `__xpg_basename()`. If the macro is undefined, `basename()` from the binary standard is used, with differences as described here:

The string identified by *`path`* shall not be modified.

If *`path`* is `"/"`, or ends with a trailing `'/'` character, the `basename()` function shall return a pointer to an empty string.

Return Value

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

See Also

`__xpg_basename()`

bind_textdomain_codeset

Name

`bind_textdomain_codeset` — specify encoding for message retrieval

Synopsis

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

Description

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

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

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

Parameters

domainname

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

codeset

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

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

Return Value

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

Errors

ENOMEM

Insufficient memory available to allocate return value.

See Also

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

bindresvport

Name

`bindresvport` — bind socket to privileged IP port

Synopsis

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

Description

If the process has appropriate privilege, the `bindresvport()` function shall bind a socket to 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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 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 misadvertently selecting the wrong or non-existant directory.

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

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

Return Value

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

Errors

`ENOMEM`

Insufficient memory was available.

See Also

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

cfmakeraw

Name

`cfmakeraw` — get and set terminal attributes

Synopsis

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

Description

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

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

termios_p->c_oflag &= ~OPOST;

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

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

termios_p->c_cflag |= CS8;
```

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

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


cfsetspeed

Name

`cfsetspeed` — set terminal input and output data rate

Synopsis

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

Errors

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

See Also

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

dcngettext

Name

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

Synopsis

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

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

Description

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

Return Value

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

Errors

`dcngettext()` shall not modify the `errno` global variable.

See Also

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

dgettext

Name

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

Synopsis

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

Description

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

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

> **Note:** The usage of *`domainanme`* 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

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

dl_iterate_phdr

Name

`dl_iterate_phdr` — iterate over a program's loaded shared objects

Synopsis

```
#include <link.h>
int dl_iterate_phdr(int(*callback) (struct dl_phdr_info *, size_t,
void *), void *data);
```

Description

`dl_iterate_phdr()` allows a program to iterate over the shared objects it has loaded. The function described by the *`callback`* parameter is called once for each loaded shared object, allowing an action to be taken for each one. *`callback`* is called with three arguments which are filled in by the implementation: a pointer to a structure of type `dl_phdr_info` containing information about the shared object; an integer size of the structure; and a copy of the *`data`* argument to `dl_iterate_phdr()`. If *`callback`* returns a non-zero value, `dl_iterate_phdr()` will stop processing, even if there are unprocessed shared objects. The order of processing is unspecified.

The `dl_phdr_info` structure has the following members (note that on 64-bit architectures the types here shown as Elf32_*`type`* will instead be Elf64_*`type`*):

```
Elf32_Addr dlpi_addr;
const char *dlpi_name;
const Elf32_Phdr *dlpi_phdr;
Elf32_Half dlpi_phnum;
unsigned long long int dlpi_adds;
unsigned long long int dlpi_subs;
size_t dlpi_tls_modid;
```



```
void *dlpi_tls_data;
```

dlpi_addr contains the base address of the shared object.

dlpi_name is a null-terminated string giving the pathname from which the shared object was loaded.

dlpi_phdr is a pointer to an array of program headers for this shared object, while *dlpi_phnum* is the number of entries in this array.

dlpi_adds and *dlpi_subs* are incremented when shared objects are added or removed, respectively.

dlpi_tls_modid contains the module ID used in TLS relocations, if there is a PT_TLS segment. Otherwise the value shall be zero.

dlpi_tls_data contains the address of the calling thread's instance of this module's PT_TLS segment, if there is one and it has been allocated in the calling thread. Otherwise the value shall be a null pointer.

Some implementations may not provide all fields in dl_phdr_info, although the first four are always mandatory. Applications are advised to have the callback function check the size parameter before examining the later members.

Return Value

The dl_iterate_phdr() function returns whatever value was returned by the last call to *callback*. This will be zero if processing completed normally, since processing does not continue unless the callback function returns zero.

Errors

No errors are defined by dl_iterate_phdr(); as noted the callback function must use a zero return to indicate success but may assign any meaning it wishes to non-zero returns.

dngettext

Name

dngettext — perform lookup in message catalog for the current locale

Synopsis

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

Description

dngettext() shall be equivalent to a call to

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

See dcngettext() for more information.

See Also

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

drand48_r

Name

`drand48_r` — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis

```
#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.

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.

envz_add, envz_entry, envz_get, envz_merge, envz_remove, envz_strip

Name

`envz_add, envz_entry, envz_get, envz_merge, envz_remove, envz_strip` — Operate on environment vectors

Synopsis

```
#include <envz.h>
error_t envz_add(char ** envz, size_t * envz_len, const char * name,
const char * value);
char envz_entry(const char * envz, size_t envz_len, const char * name);
char envz_get(const char * envz, size_t envz_len, const char * name);
error_t envz_merge(char ** envz, size_t * envz_len, const char * envz2,
size_t envz2_len, int override);
void envz_remove(char ** envz, size_t * envz_len, const char * name);
void envz_strip(char ** envz, size_t * envz_len);
```

Description

The envz functions operate on envz vectors, which are typically used to manipulate program environment variables.

An envz vector is identical in makeup to an argz vector (see argz_add, argz_add_sep, argz_append, argz_count, argz_create, argz_create_sep, argz_delete, argz_extract, argz_insert, argz_next, argz_replace, argz_stringify) but has the constraint that each element is a name, value pair separated by an = character. Only the first = character in an element has special meaning, any subsequent instances are part of the value string. If no = character is present in an element, the value is taken to be `NULL`. If an element has an empty value (an = character is present), the value will return the empty string "" when queried.

Since an envz vector is an argz vector, the argz functions can be used where it makes sense. For example, converting from a program's environment variables (as described in Chapter 8 of the XBD volume of POSIX 1003.1-2008 (ISO/IEC 9945-2009)) to an envz vector is done with `argz_create()`.

The `envz_add()` function adds a string constructed from `name` and `value` in the form "`name=value`" to the envz vector identified by `envz` and `envz_len`, updating `envz` and `envz_len`. If `value` is `NULL` it adds a string of the form "`name`". If an entry with the same name already exists, it is replaced..

The `envz_entry()` function searches for `name` in the envz vector identified by `envz` and `envz_len`, returning the full entry if found, or `NULL` if not.

The `envz_get()` function searches for `name` in the envz vector identified by `envz` and `envz_len`, returning the value part of the entry if found, or `NULL` if not. Note the value may be also `NULL`.

The `envz_merge()` function adds each entry from the envz vector identified by `envz2` and `envz2_len` to the envz vector identified by `envz` and `envz_len`, updating `envz` and `envz_len`. The behavior is as if `envz_add()` were called for each entry in `envz2`. If `override` is true, then values from `envz2` will replace those with the same name in `envz`.

The `envz_remove()` function removes the entry for `name` from the envz vector identified by `envz` and `envz_len` if it exists, updating `envz` and `envz_len`.

The `envz_strip()` function removes all entries with value `NULL`.

Return Value

The envz functions that perform memory allocation (`envz_add()` and `envz_merge()`) return an error_t type. These functions return `0` on success; if memory allocation fails, they return ENOMEM.

`envz_entry()` and `envz_get()` return a pointer to a substring in an envz vector, or `NULL`.

See Also

argz_add, argz_add_sep, argz_append, argz_count, argz_create, argz_create_sep, argz_delete, argz_extract, argz_insert, argz_next, argz_replace, argz_stringify

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

```
#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 <error.h>
void error (int status , 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 *`status`* is nonzero, `error()` shall call `exit(status)`.

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with differences as listed below.

Implementation may set `O_LARGEFILE`

According to POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 *or*ing) along with one of the other operations.

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

Return Value

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

Errors

EWOULDBLOCK

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

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.

fnmatch

Name

`fnmatch` — match a filename or a pathname

Description

`fnmatch()` is as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with differences as listed below.

Additional flags

In addition to the available values that can be used to form *`flags`*, as documented in POSIX 1003.1-2008 (ISO/IEC 9945-2009), this specification permits the following constants.

`FNM_CASEFOLD`

If this flag is set, the pattern is matched case-insensitively.

`FNM_FILE_NAME`

A synonym for `FNM_PATHNAME` constant specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

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()`.

fscanf

Name

`fscanf` — convert formatted input

Description

The `scanf()` family of functions shall behave as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 idenfitied 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 idenfitied by *`buf`* accordingly. On error, the `fstatfs64()` function shall return -1 and set `errno` accordingly.

Errors

See `fstatfs()`.

futimes

Name

`futimes, lutimes` — set file access and modification times

Synopsis

```
#include <sys/time.h>
int futimes(int fd, const struct timeval tv[2]);
int lutimes(const char * filename, const struct timeval tv[2]);
```

Description

The `futimes()` and `lutimes()` functions shall set the access and modification times of a file to the values of the `tv` argument, which is an array of two timeval structures. The behavior is as for `utimes()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

The `futimes()` function shall change the times of of the open file described by file descriptor `fd`.

The `lutimes()` function shall change the times of of the file pointed to by the `filename` argument, except that if `filename` refers to a symbolic link, then the link is not followed and the times of the symbolic link are changed. This is similar to supplying `AT_SYMLINK_NOFOLLOW` in the `flag` argument to the `utimensat()` function.

Errors

As for `utimes()`, but in addition:

ENOSYS

Ths implementation does not support this function (for `lutimes()`).

The implementation could not access a resource needed to complete the function (for `futimes()`).

See Also

`utimes(), utime(), utimensat().`

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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.

getcwd

Name

getcwd — get the pathname of the current working directory

Synopsis

```
#include <unistd.h>
char * getcwd(char * buf, size_t size);
```

Description

The getcwd() functions shall behave as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009), except as noted below.

Differences

If buf is NULL, memory is allocated for buf. If size is 0, the allocation size will be the size of the pathname +1, else the requested size is allocated.

Changed or Added Errors

EINVAL

The size argument is 0 and buf is not a null pointer.

ENOENT

The current working directory has been unlinked.

getdomainname

Name

getdomainname — get NIS domain name (DEPRECATED).

Synopsis

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

Description

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

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

Return Value

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

Errors

EINVAL

name is a null pointer.

EINVAL

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

Future Directions

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

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 interface `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 supplementary 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 actually 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 supplementary 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 returning `-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

`getgroups()`

gethostbyaddr_r

Name

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

Synopsis

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

Description

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

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

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

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

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

Return Value

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

gethostbyname2

Name

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

Synopsis

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

Description

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

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

Return Value

On success, the `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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

Errors

The `gethostbyname2()` shall set `h_errno` as for `gethostbyname()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

gethostbyname2_r

Name

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

Synopsis

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

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 `gethostbyname2()`. These functions are similar to the `gethostbyname()` and `gethostbyname_r()` functions but additionally allow the search to be restricted to a particular address family specified by *af*.

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

Return Value

On success, the `gethostbyname2_r()` function shall return zero. If the return value was ERANGE, the size of the buffer *buf*, indicated by *buflen*, was too small. If the `gethostbyname2_r()` function returns returns any other value, then the variable referenced by *h_errnop* 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()`.

getifaddrs

Name

`getifaddrs, freeifaddrs` — get interface addresses

Synopsis

```
#include <ifaddrs.h>
int getifaddrs(struct ifaddrs ** ifap);
void freeifaddrs(struct ifaddrs * ifa);
```

Description

The `getifaddrs()` function creates a linked list of structures describing the network interfaces of the local system. The address of the first item is stored in memory pointed to by `ifap`. The data returned is dynamically allocated, and should be freed using `freeifaddrs()`.

The list consists of structures of type `ifaddrs` (see Data Definitions above).

Return Value

On success, `getifaddrs()` returns zero; on error, `-1` is returned, and `errno` is set appropriately.

Errors

`getifaddrs()` may fail and set errno for any of the errors specified for `socket()`, `bind()`, `getsockname()`, `recvmsg()`, `sendto()`, `malloc()`, or `realloc()`.

See Also

`bind()`, `getsockname()`, `socket()`.

getloadavg

Name

`getloadavg` — get system load averages

Synopsis

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

Description

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

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

```
#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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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.

getopt_long

Name

`getopt_long` — parse command line options

Synopsis

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

Description

`getopt_long()` works like `getopt()` except that it also accepts long options, started out by two dashes. Long option names may be abbreviated if the abbreviation is unique or is an exact match for some defined option. A long option may take a parameter, of the form `--arg=param` or `--arg param`.

`longopts` is a pointer to the first element of an array of struct `option` declared in `getopt.h` as:

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



```
};
```

The fields in this structure have the following meaning:

name

The name of the long option.

has_arg

One of:

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

flag

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

val

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

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

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

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

getopt_long_only

Name

`getopt_long_only` — parse command line options

Synopsis

```
#define _GNU_SOURCE
```

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

Description

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

> **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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

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 meory 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.

getprotobyname_r

Name

`getprotobyname_r` — retrieve information from the network protocol database by protocol name, reentrantly

Synopsis

```
#include <netdb.h>
int getprotobyname_r(const char * name, struct protoent * result_buf,
char * buf, size_t buflen, struct protoent * * result);
```

Description

The `getprotobyname_r()` function is a reentrant version of the `getprotobyname()` function.

The `getprotobyname_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 `getprotobyname_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

```
#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.

getrlimit, setrlimit

Name

`getrlimit, setrlimit` — get resource consumption limits

Synopsis

```
#include <sys/resource.h>
int getrlimit(__rlimit_resource_t __resource, struct rlimit *
__rlimits);
int setrlimit(__rlimit_resource_t __resource, const struct rlimit *
__rlimits);
```

Description

`getrlimit()` and `setrlimit()` are as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with differences as listed below.

Extra Resources

These additional resources extend the list in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

RLIMIT_NPROC

The maximum number of processes (or, more precisely on Linux, threads) that can be created for the real user ID of the calling process. Upon encountering this limit, `fork()` shall fail with the error EAGAIN.

RLIMIT_MEMLOCK

The maximum number of bytes of memory that may be locked into RAM. In effect this limit is rounded down to the nearest multiple of the system page size. This limit affects `mlock()` and `mlockall()`, the `mmap()` `MAP_LOCKED` operation and the `shmctl()` `SHM_LOCK` operation. The `shmctl()` `SHM_LOCK` locks are accounted for separately from the per-process memory locks established by `mlock()`, `mlockall()`, and `mmap()` `MAP_LOCKED`. In the former case, the limit sets a maximum on the total bytes in shared memory segments (see `shmget()`) that may be locked by the real user ID of the calling process. A process can lock bytes up to this limit in each of these two categories.

RLIMIT_LOCKS

A limit on the combined number of `flock()` locks and `fcntl()` leases that this process may establish. This limit is obsolete and should not be used; support depends heavily on the version of the operating system kernel.

RLIMIT_RSS

Specifies the limit (in pages) of the process's resident set. This limit is obsolete and should not be used; support depends heavily on the version of the operating system kernel. It affects only calls to `madvise()` specifying `MADV_WILLNEED`.

RLIMIT_SIGPENDING

Specifies the limit on the number of signals that may be queued for the real user ID of the calling process. Both standard and real-time signals are counted for the purpose of checking this limit. However, the limit is enforced only for `sigqueue()`; it is always possible to use `kill()` to queue one instance of any of the signals that are not already queued to the process.

RLIMIT_MSGQUEUE

Specifies the limit on the number of bytes that can be allocated for POSIX message queues for the real user ID of the calling process. This limit is enforced for `mq_open()`. Each message queue that the user creates counts (until it is removed) against this limit according to the formula:

```
bytes = attr.mq_maxmsg * sizeof(struct msg_msg *) +
```



```
attr.mq_maxmsg * attr.mq_msgsize
```

where attr is the mq_attr structure specified as the fourth argument to mq_open(3).

The first addend in the formula, which includes sizeof(struct msg_msg *) (4 bytes on Linux/i386), ensures that the user cannot create an unlimited number of zero-length messages (such messages nevertheless each consume some system memory for bookkeeping overhead).

RLIMIT_NICE

Specifies a ceiling to which the process's nice value can be raised using `setpriority()` or `nice()`. The actual ceiling for the nice value is calculated as 20 minus the value of `rlim_cur`.

RLIMIT_RTPRIO

Specifies a ceiling on the real-time priority that may be set for this process using sched_setscheduler(2) and sched_setparam(2).

RLIMIT_RTTIME

Specifies a limit (in microseconds) on the amount of CPU time that a process scheduled under a real-time scheduling policy may consume without making a blocking system call. For the purpose of this limit, each time a process makes a blocking system call, the count of its consumed CPU time is reset to zero. The CPU time count is not reset if the process continues trying to use the CPU but is preempted, its time slice expires, or it calls `sched_yield()`.

Upon reaching the soft limit, the process is sent a `SIGXCPU` signal. If the process catches or ignores this signal and continues consuming CPU time, then `SIGXCPU` will be generated once each second until the hard limit is reached, at which point the process is sent a `SIGKILL` signal.

The intended use of this limit is to stop a runaway real-time process from locking up the system.

Extra Errors

These additional error codes extend the list in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

EFAULT

A pointer argument points to a location outside the accessible address space.

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

`getservbyport_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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*, 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). It shall search for pathnames matching *`pattern`* according to the rules used by the shell, `/bin/sh`. No tilde expansion or parameter substitution is done; see `wordexp()`.

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

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



```
glob64_t;
```

Structure members with the same name as corresponding members of a `glob_t` as defined in POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

gnu_get_libc_version, gnu_get_libc_release

Name

`gnu_get_libc_version, gnu_get_libc_release` — get glibc-specific version and release

Synopsis

```
#include <gnu/libc-version.h>
const char * gnu_get_libc_version(void);
const char * gnu_get_libc_release(void);
```

Description

`gnu_get_libc_version()` returns a string that identifies the version of the C library running the program making the call.

`gnu_get_libc_release()` returns a string indicates the release status of the C library running the program making the call. This will be a string such as `"stable"`.

Return Value

The functions return strings. The contents of these strings are unspecified.

Errors

No errors are defined.

Notes

These functions are specific to GNU libc (glibc). This specification does not require the implementation of libc to be glibc, although it requires these functions.

The string returned by `gnu_get_libc_version()` will be a dotted version string, which may have meaning to developers otherwise familiar with glibc. These functions have been requested to aid in portability of software which also runs in non-LSB contexts, but decisions based on the return value should be tempered by an understanding of what the behavioral requirements of this specification are. That is, it may or may not be useful to discover that a running system, for example, has version `"2.10.1"` if that implies different behavior than described by this specification.

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 containing 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 initialized 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 anet_iton(const char * cp, struct in_addr * inp);
```

Description

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

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

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

initgroups

Name

`initgroups` — initialize the supplementary group access list

Synopsis

```
#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:

```
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

Watched 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, int 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) contains an interface named `ioctl()`. The LSB only defines behavior when *`fildes`* refers to a socket (see sockio) or terminal device (see ttyio), while POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009)). 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.

`SIOCGIFBRDADDR`

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:

`TIOCGWINSZ`

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

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

`TIOCSWINSZ`

Sets the size attributes of the terminal or pseudo-terminal identified by *fd*. On entry, *argp* shall reference a `winsize` structure. The value of the `winsize` structure's element *ws_row* shall be the number of rows of text (i.e. lines of text) that can be viewed on the device, and the element *ws_col* shall be the number of columns (i.e. text width). Note that this call merely sets the size attributes for the kernel driver, not the window size itself, and is intended to be used to update the kernel driver when the window size is changed.

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

```
#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

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

Description

`kill()` is as specified in the *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*, 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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)* 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 <stdlib.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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*, except with differences as listed below.

Need Not Follow Symlinks

POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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.

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 unsigned char) in the initial *n* bytes (each interpreted as an unsigned char) of the object pointed to by *s*.

Return Value

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

Errors

No errors are defined.

See Also

`memchr()`

mkstemp64

Name

`mkstemp64` — create a unique temporary file (Large File Support)

Synopsis

```
#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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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

```
#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.

ngettext

Name

`ngettext` — search message catalogs for plural string

Synopsis

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

Description

The `ngettext()` function shall search the currently selected message catalogs for a string matching the singular string *`msgid1`*. If a string is located, and if *`n`* is 1, that string shall be returned. If *`n`* is not 1, a pluralized version (dependent on *`n`*) of the string shall be returned.

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

Return Value

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

Applications shall not modify the string returned by `ngettext()`.

Errors

None.

The `ngettext()` function shall not modify `errno`.

See Also

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

nrand48_r

Name

nrand48_r — reentrantly generate pseudorandom numbers in a uniform distribution

Synopsis

```
#include <stdlib.h>
int nrand48_r(unsigned short[3] xsubi, struct drand48_data * buffer,
long int * result);
```

Description

The interface nrand48_r() shall function in the same way as the interface nrand48(), except that nrand48_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 jrand48_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

```
#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 (ISO/IEC 9945-2009). 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 *`[program,version,protocol]`* and *`port`* on the machine's RPC Bind service. The value of *`protocol`* is most likely `IPPROTO_UDP` or `IPPROTO_TCP`. Automatically done by `svc_register()`.

Return Value

`pmap_set()` returns non-zero if it suceeds, 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 [*`prognum`*,*`versnum`*, *] and `ports` on the machine's RPC Bind service.

Return Value

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

posix_fadvise64

Name

`posix_fadvise64` — File advisory information (Large File Support)

Synopsis

```
#include <fcntl.h>
int posix_fadvise64(int fd, off64_t offset, off64_t len, int advice);
```

Description

The `posix_fadvise64()` function is a large-file version of the `posix_fadvise()` function defined in POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), except that the *`offset`* and *`len`* arguments are `off64_t` objects rather than `off_t`.

Return Value

See `posix_fallocate()`.

Errors

See `posix_fallocate()`.

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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.

ptrace

Name

`ptrace` — process trace

Synopsis

```
#include <sys/ptrace.h>
long ptrace(enum __ptrace_request request, pid_t pid, void * addr,
void * data);
```

Description

The `ptrace()` system call shall enable a process to observe and control the execution of another process, as well as examine and change certain attributes of that process.

This function operates via requests, which act on the traced process using the other parameters in ways unique to each request type. The tracing process must initiate tracing, either via the `PTRACE_TRACEME` or `PTRACE_ATTACH` requests, before other requests may be performed. Except for `PTRACE_TRACEME` and `PTRACE_KILL`, all requests must be performed on a traced process that has been stopped.

All signals, except one, delivered to the traced process cause it to stop, irrespective of its registered signal handling, and cause an event to be delivered to the tracing process which can be detected using the wait(2) system call. The exception is the `SIGKILL` signal, which is delivered immediately and performs its usual specified behavior.

The following requests are defined:

`PTRACE_TRACEME`

This request initates a trace from the perspective of the traced process, indicating that the parent of the current process shall be the tracing process. When this is called, a subsequent call to execve(2) shall cause the tracing process to receive a SIGTRAP signal, and shall stop the current process. This is the only request a traced process may perform, and a tracing process may not perform this request. The other parameters are ignored.

`PTRACE_ATTACH`

This request initates a trace from the perspective of the tracing process on the process specified by *`pid`*. After this call succeeds, the traced process will appear to be a child of the tracing process, although the original parent will still be returned to the traced process via getppid(2). The traced process will receive a `SIGSTOP` signal; the tracing process should use wait(2) to ensure that the traced process has stopped. A tracing process is only guaranteed to be able to trace its child processes; the tracing of other processes may not be allowed by the system, and the process with process ID 1 may not be traced under any circumstances. The *`addr`* and *`data`* parameters are ignored.

`PTRACE_CONT`

This request restarts a traced process, given in *pid*, which has been stopped. The *data* parameter may point to a signal ID to deliver to the traced process; if it is zero or SIGSTOP, no signal is delivered to the child. The *addr* is ignored.

PTRACE_DETACH

This request performs the same function, in the same way, as PTRACE_CONT, except that the tracing relationship between the tracing and traced processes is also undone. If the trace was initiated using PTRACE_ATTACH, the original parent-child relationships that existed beforehand are restored.

PTRACE_KILL

This request causes a SIGKILL signal to be sent to the traced process specified in *pid*. The *addr* and *data* parameters are ignored.

PTRACE_PEEKTEXT

This request reads a word at the location *addr* of the traced process *pid*, and returns it to the caller. The *data* parameter is ignored.

PTRACE_PEEKDATA

This request performs identically to the PTRACE_PEEKTEXT request.

PTRACE_PEEKUSER

This request reads a word at offset *addr* in the USER area of the traced process *pid*. The offset must be word-aligned. The *data* parameter is ignored.

PTRACE_POKETEXT

This request writes the word pointed at by *data* to the location *addr* of the traced process *pid*.

PTRACE_POKEDATA

This request performs identically to the PTRACE_POKETEXT request.

PTRACE_POKEUSER

This request writes the word pointed at by *data* to offset *addr* in the USER area of the traced process *pid*. The offset must be word-aligned. Implementations may choose to disallow some modifications to the USER area.

PTRACE_GETREGS

This request copies the general purpose registers from the traced process *pid* to the tracing process at location *data*. This parameter may not be available on all architectures. The *addr* parameter is ignored.

PTRACE_GETFPREGS

This request copies the floating point registers from the traced process *pid* to the tracing process at location *data*. This parameter may not be available on all architectures. The *addr* parameter is ignored.

PTRACE_SETREGS

This request writes the general purpose registers to the traced process *pid* from the tracing process at location *data*. This parameter may not be available on all architectures. Implementations may choose to disallow some register modifications. The *addr* parameter is ignored.

PTRACE_SETFPREGS

This request writes the floating point registers to the traced process *pid* from the tracing process at location *data*. This parameter may not be available on all architectures. Implementations may choose to disallow some register modifications. The *addr* parameter is ignored.

PTRACE_GETSIGINFO

This request writes information about the signal which caused the traced process *pid* to stop to the tracing process at location *data*, as a siginfo_t. The *addr* parameter is ignored.

PTRACE_SETSIGINFO

This request writes signal information to the traced process *pid* from a siginfo_t structure pointed at by *data*, such that it will be used as the signal information by the traced process when it is resumed. The *addr* parameter is ignored.

PTRACE_GETEVENTMSG

This request stores information about the most recent ptrace event for the traced process *pid* in the unsigned long pointed at by *data*. For PTRACE_EVENT_EXIT, this is the exit status of the traced process. For PTRACE_EVENT_FORK, PTRACE_EVENT_VFORK, or PTRACE_EVENT_CLONE, this is the PID of the newly created process. The *addr* parameter is ignored.

PTRACE_SYSCALL

This request performs the same function, in the same way, as PTRACE_CONT, but with the additional step of causing the traced process to stop at the next entry to or exit from a system call. The usual events that would also cause the traced process to stop continue to do so.

PTRACE_SINGLESTEP

This request performs the same function, in the same way, as PTRACE_CONT, but with the additional step of causing the traced process to stop after execution of a single instruction. The usual events that would also cause the traced process to stop continue to do so.

PTRACE_SYSEMU

This request performs the same function, in the same way, as PTRACE_CONT, but with the additional step of causing the traced process to stop on entry to the next syscall, which will then not be executed.

PTRACE_SYSEMU_SINGLESTEP

This request performs the same function, in the same way, as PTRACE_CONT, but with the additional step of causing the traced process to stop on entry to the next syscall, which will then not be executed. If the next instruction is not itself a syscall, the traced process will stop after a single instruction is executed.

PTRACE_SETOPTIONS

This request sets ptrace() options for the traced process *pid* from the location pointed to by *data*. The *addr* is ignored. This location is interpreted as a bitmask of options, as defined by the following flags:

PTRACE_O_TRACESYSGOOD

This option, when set, causes syscall traps to set bit 7 in the signal number.

PTRACE_O_TRACEFORK

This option, when set, causes the traced process to stop when it calls fork(2). The original traced process will stop with SIGTRAP | PTRACE_EVENT_FORK << 8, and the new process will be stopped with SIGSTOP. The new process will also be traced by the tracing process, as if the tracing process had sent the PTRACE_ATTACH request for that process. The PID of the new process may be retrieved with the PTRACE_GETEVENTMSG request.

PTRACE_O_TRACEVFORK

This option, when set, causes the traced process to stop when it calls vfork(2). The original traced process will stop with SIGTRAP | PTRACE_EVENT_VFORK << 8, and the new process will be stopped with SIGSTOP. The new process will also be traced by the tracing process, as if the tracing process had sent the PTRACE_ATTACH request for that process. The PID of the new process may be retrieved with the PTRACE_GETEVENTMSG request.

PTRACE_O_TRACECLONE

This option, when set, causes the traced process to stop when it calls clone(2). The original traced process will stop with SIGTRAP | PTRACE_EVENT_CLONE << 8, and the new process will be stopped with SIGSTOP. The new process will also be traced by the tracing process, as if the tracing process had sent the PTRACE_ATTACH request for that process. The PID of the new process may be retrieved with the PTRACE_GETEVENTMSG request. Under certain circumstances, clone(2) calls by the traced process will generate events and information consistent with the PTRACE_O_TRACEVFORK or PTRACE_O_TRACEFORK options above.

PTRACE_O_TRACEEXEC

This option, when set, causes the traced process to stop when it calls execve(2). The traced process will stop with SIGTRAP | PTRACE_EVENT_EXEC << 8.

PTRACE_O_TRACEVFORKDONE

This option, when set, causes the traced process to stop at the completion of its next vfork(2) call. The traced process will stop with SIGTRAP | PTRACE_EVENT_EXEC << 8.

`PTRACE_O_TRACEEXIT`

This option, when set, causes the traced process to stop upon exit. The traced process will stop with SIGTRAP | PTRACE_EVENT_EXIT << 8, and its exit status can be retrieved with the PTRACE_GETEVENTMSG request. The stop is guaranteed to be early in the process exit process, meaning that information such as register status at exit is preserved. Upon continuing, the traced process will immediately exit.

Return Value

On success, `ptrace()` shall return the requested data for `PTRACE_PEEK` requests, or zero for all other requests. On error, all requests return -1, with `errno` set to an appropriate value. Note that -1 may be a valid return value for `PTRACE_PEEK` requests; the application is responsible for distinguishing between an error condition and a valid return value in that case.

Errors

On error, `ptrace()` shall set `errno` to one of the regular error values below:

`EBUSY`

An error occurred while allocating or freeing a debug register.

`EFAULT`

The request attempted to read from or write to an invalid area in the memory space of the tracing or traced process.

`EIO`

The request was invalid, or it attempted to read from or write to an invalid area in the memory space of the tracing or traced process, or it violated a word-alignment boundary, or an invalid signal was given to continue the traced process.

`EINVAL`

An attempt was made to set an invalid option.

`EPERM`

The request to trace a process was denied by the system.

`ESRCH`

The process requested does not exist, is not being traced by the current process, or is not stopped.

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()`.

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009). 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. Otherwise `pwrite()` shall return -1 and set `errno` to indicate the error.

Errors

See `pwrite()` for possible error values.

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*, 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

```
#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 (ISO/IEC 9945-2009). 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), except as noted below.

Differences

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

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

sched_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 example, 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 prevents 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 *`in_fd`*.

Notes

`sendfile()` is usually faster than combining `read()` and `write()` calls, because it is part of the kernel. However, if it fails with `EINVAL`, falling back to `read()` and `write()` may be advisable.

It is advisable for performance reasons to use the `TCP_CORK` option of the `tcp()` function when sending header data with file contents to a TCP socket. This minimizes the number of packets.

See Also

`mmap()`, `open()`, `socket()`, `splice()`.

sendfile64

Name

`sendfile64` — transfer data between two file descriptors (Large File Support)

Synopsis

```
#include <sys/sendfile.h>
ssize_t sendfile64(int out_fd, int in_fd, off64_t * offset, size_t
count);
```

Description

The `sendfile64()` function is a large-file version of the `sendfile()` function.

setbuffer

Name

`setbuffer` — stream buffering operation

Synopsis

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

Description

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

setgroups

Name

`setgroups` — set list of supplementary group IDs

Synopsis

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

Description

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

Return Value

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

Errors

EFAULT

`list` has an invalid address.

EPERM

The process does not have appropriate privileges.

EINVAL

`size` is greater than `NGROUPS_MAX`.

sethostname

Name

`sethostname` — set host name

Synopsis

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

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

Description

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

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

Return Value

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

Errors

EINVAL

len is negative or larger than the maximum allowed size.

EPERM

the process did not have appropriate privilege.

EFAULT

name is an invalid address.

Rationale

POSIX 1003.1-2008 (ISO/IEC 9945-2009) guarantees that:

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

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

setsockopt

Name

`setsockopt` — set 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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*, 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`

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 either an `ip_mreqn` structure or an `in_addr` structure. If using the `ip_mreqn` structure, the *`imr_multiaddr`* field should be set to multicast group address, the *`imr_address`* field to the address of the local interface, and the *`imr_index`* field to the interface index. If using the `in_addr` structure, the address of the local interface shall be specified. If *`in_addr`* or *`imr_address`* is set to `INADDR_ANY`, then an appropriate interface is chosen by the system. If *`imr_index`* is zero, then an appropriate interface index is chosen by the implementation.

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 POSIX 1003.1-2001 (ISO/IEC 9945-2003).

setstate_r

Name

`setstate_r` — reentrantly change the state array used by random number generator functions

Synopsis

```
#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

```
#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

```
#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

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

Description

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

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

Return Value

The `sigisemptyset()` function shall return a positive non-zero value if the signal set referenced by *`set`* is empty, or zero if this set is empty. There are no defined error returns.

sigorset

Name

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

Synopsis

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

Description

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

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

Return Value

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

See Also

`sigandset()`

sigpause

Name

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

Synopsis

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

Description

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

In the source standard, `sigpause()` is implemented as a macro causing it to behave as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 idenfitied 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

```
#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 idenfitied by *buf* accordingly. On error, the `statfs64()` function shall return -1 and set `errno` accordingly.

Errors

See `fstatfs()`.

stime

Name

`stime` — set time

Synopsis

```
#define _SVID_SOURCE
```

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

Description

If the process has appropriate privilege, the `stime()` function shall set the system's idea of the time and date. Time, referenced by *t*, is measured in seconds from the epoch (defined in POSIX 1003.1-2008 (ISO/IEC 9945-2009) 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.

strcasestr

Name

`strcasestr` — locate a substring ignoring case

Synopsis

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

Description

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

Return Value

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

strerror_r

Name

`strerror_r` — 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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()`

strptime

Name

`strptime` — parse a time string

Description

The `strptime()` shall behave as specified in the *POSIX 1003.1-2008 (ISO/IEC 9945-2009)* with differences as listed below.

Number of leading zeroes may be limited

The *POSIX 1003.1-2008 (ISO/IEC 9945-2009)* 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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)* 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) for this matter.

strsep

Name

`strsep` — extract token from string

Synopsis

```
#include <string.h>
char * strsep(char * * stringp, const char * delim);
```

Description

The `strsep()` function shall find the first token in the string referenced by the pointer *`stringp`*, using the characters in *`delim`* as delimiters.

If *`stringp`* is NULL, `strsep()` shall return NULL and do nothing else.

If *`stringp`* is non-NULL, `strsep()` shall find the first token in the string referenced by *`stringp`*, where tokens are delimited by characters in the string *`delim`*. This token shall be terminated with a `\0` character by overwriting the delimiter, and *`stringp`* shall be updated to point past the token. In case no delimiter was found, the token is taken to be the entire string referenced by *`stringp`*, and the location referenced by *`stringp`* is made NULL.

Return Value

`strsep()` shall return a pointer to the beginning of the token.

Notes

The `strsep()` function was introduced as a replacement for `strtok()`, since the latter cannot handle empty fields. However, `strtok()` conforms to ISO C (1999) and to POSIX 1003.1-2008 (ISO/IEC 9945-2009) and hence is more portable.

See Also

`strtok()`, `strtok_r()`.

strtoq

Name

`strtoq` — convert string value to a long or quad_t integer

Synopsis

```
#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 quadt value. The conversion is done according to the given base, which shall be between `2` and `36` inclusive, or be the special value `0`.

`nptr` may begin with an arbitrary amount of white space (as determined by `isspace()`), 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

```
#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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 "`%a`seconds" will have a different meaning on an LSB conforming system.

sysconf

Name

`sysconf` — Get configuration information at runtime

Synopsis

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

DESCRIPTION

`sysconf()` is as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with differences as listed below.

Extra Variables

These additional values extend the list in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

- *_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).

Extra Versions

While this specification only requires conformance with POSIX 1003.1-2008 (ISO/IEC 9945-2009), implementations are not constrained from moving on and claiming conformance with a subsequent edition, POSIX 1003.1-2008 (ISO/IEC 9945-2009). Thus for run-time checks using `sysconf()`, the wording is amended to allow return values of `0`, `-1`, `200112L` or `200809L` where formerly `200809L` was not listed as allowed.

sysinfo

Name

`sysinfo` — return system information

Synopsis

```
#include <sys/sysinfo.h>
int sysinfo(struct sysinfo *info);
```

Description

`sysinfo()` provides a way to obtain certain system statistics. Statistics are written into a `sysinfo` structure pointed to by *`info`*. Elements which take a size are sized in units indicated by the value of the *`mem_unit`* member of *`info`*. The other members have traditional meanings as indicated in Data Definitions, but are not formally part of this specification.

Return Value

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

Errors

EFAULT

The *`info`* parameter does not point to a valid `sysinfo` structure.

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

Notes

The fact that `system()` ignores interrupts is often not what a program wants. POSIX 1003.1-2008 (ISO/IEC 9945-2009) describes some of the consequences; an additional consequence is that a program calling `system()` from a loop cannot be reliably interrupted. Many programs will want to use the `exec()` family of functions instead.

Do not use `system()` from a program with `suid` or `sgid` privileges, because unexpected values for some environment variables might be used to subvert system integrity. Use the `exec()` family of functions instead, but not `execlp()` or `execvp()`. `system()` will not, in fact, work properly from programs with `suid` or `sgid` privileges on systems on which `/bin/sh` is **bash** version 2, since **bash** 2 drops privileges on startup. (Debian uses a modified **bash** which does not do this when invoked as **sh**.)

The check for the availability of `/bin/sh` is not actually performed; it is always assumed to be available. ISO C (1999) specifies the check, but POSIX 1003.1-2008 (ISO/IEC 9945-2009) specifies that the return shall always be nonzero, since a system without the shell is not conforming, and it is this that is implemented.

It is possible for the shell command to return 127, so that code is not a sure indication that the `execve()` call failed; check the global variable `errno` to make sure.

textdomain

Name

`textdomain` — set the current default message domain

Synopsis

```
#include <libintl.h>
char * textdomain(const char * domainname);
```

Description

The `textdomain()` function shall set the current default message domain to *domainname*. Subsequent calls to `gettext()` and `ngettext()` use the default message domain.

If *domainname* is NULL, the default message domain shall not be altered.

If *domainname* is "", `textdomain()` shall reset the default domain to the system default of "messages".

Return

On success, `textdomain()` shall return the currently selected domain. Otherwise, a null pointer shall be returned, and `errno` is set to indicate the error.

Errors

`ENOMEM`

Insufficent memory available.

unlink

Name

`unlink` — remove a directory entry

Synopsis

```
int unlink(const char * path);
```

Description

`unlink()` is as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

Rationale: The Linux kernel has deliberately chosen EISDIR for this case and does not expect to change.

utmpname

Name

utmpname — set user accounting database

Synopsis

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

Description

The utmpname() function shall cause the user accounting database used by the getutent(), getutent_r(), getutxent(), getutxid(), getutxline(), and pututxline() functions to be that named by *dbname*, instead of the system default database. See Section 18.3 for further information.

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

Return Value

None.

Errors

None defined.

vasprintf

Name

vasprintf — write formatted output to a dynamically allocated string

Synopsis

```
#include <stdarg.h>
#include <stdio.h>
int vasprintf(char * * restrict ptr, const char * restrict format,
va_list arg);
```

Description

The vasprintf() function shall write formatted output to a dynamically allocated string, and store the address of that string in the location referenced by *ptr*. It shall behave as asprintf(), except that instead of being called with a variable number of arguments, it is called with an argument list as defined by <stdarg.h>.

Return Value

Refer to fprintf().

Errors

Refer to fprintf().

verrx

Name

verrx — display formatted error message and exit

Synopsis

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

Description

The verrx() shall behave as errx() except that instead of being called with a variable number of arguments, it is called with an argument list as defined by <stdarg.h>.

verrx() does not return, but exits with the value of *eval*.

Return Value

None.

Errors

None.

vfscanf

Name

vfscanf — convert formatted input

Description

The scanf() family of functions shall behave as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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.

vfwscanf

Name

vfwscanf — convert formatted input

Description

The scanf() family of functions shall behave as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), except as noted below.

Differences

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

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

vsyslog

Name

vsyslog — log to system log

Synopsis

```
#include <stdarg.h>
#include <syslog.h>
void vsyslog(int priority, char * message, va_list arglist);
```

Description

The vsyslog() function is identical to syslog() as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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.

wcstoq

Name

`wcstoq` — convert wide string to long long int representation

Synopsis

```
#include <wchar.h>
long long int wcstoq(const wchar_t * restrict nptr, wchar_t **
restrict endptr, int base);
```

Description

The `wcstoq()` function shall convert the initial portion of the wide string *`nptr`* to `long long int` representation. It is identical to `wcstoll()`.

Return Value

Refer to `wcstoll()`.

Errors

Refer to `wcstoll()`.

wcstouq

Name

`wcstouq` — convert wide string to unsigned long long int representation

Synopsis

```
#include <wchar.h>
unsigned long long 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), except as noted below.

Differences

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

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

xdr_u_int

Name

`xdr_u_int` — library routines for external data representation

Synopsis

```
int xdr_u_int(XDR * xdrs, unsigned int * up);
```

Description

`xdr_u_int()` is a filter primitive that translates between C unsigned integers and their external representations.

Return Value

On success, 1 is returned. On error, 0 is returned.

xdrstdio_create

Name

xdrstdio_create — library routines for external data representation

Synopsis

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

Description

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

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

Return Value

None.

14.6 Interfaces for libm

Table 14-38 defines the library name and shared object name for the libm library

Table 14-38 libm Definition

Library:	libm
SONAME:	See architecture specific part.

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

[LSB] This Specification
[SUSv3] POSIX 1003.1-2001 (ISO/IEC 9945-2003)
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

14.6.1 Math

14.6.1.1 Interfaces for Math

An LSB conforming implementation shall provide the generic functions for Math specified in Table 14-39, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-39 libm - Math Function Interfaces

__finite [LSB]	__finitef [LSB]	__finitel [LSB]	__fpclassify [LSB]
__fpclassifyf [LSB]	__signbit [LSB]	__signbitf [LSB]	acos [SUSv4]

acosf [SUSv4]	acosh [SUSv4]	acoshf [SUSv4]	acoshl [SUSv4]
acosl [SUSv4]	asin [SUSv4]	asinf [SUSv4]	asinh [SUSv4]
asinhf [SUSv4]	asinhl [SUSv4]	asinl [SUSv4]	atan [SUSv4]
atan2 [SUSv4]	atan2f [SUSv4]	atan2l [SUSv4]	atanf [SUSv4]
atanh [SUSv4]	atanhf [SUSv4]	atanhl [SUSv4]	atanl [SUSv4]
cabs [SUSv4]	cabsf [SUSv4]	cabsl [SUSv4]	cacos [SUSv4]
cacosf [SUSv4]	cacosh [SUSv4]	cacoshf [SUSv4]	cacoshl [SUSv4]
cacosl [SUSv4]	carg [SUSv4]	cargf [SUSv4]	cargl [SUSv4]
casin [SUSv4]	casinf [SUSv4]	casinh [SUSv4]	casinhf [SUSv4]
casinhl [SUSv4]	casinl [SUSv4]	catan [SUSv4]	catanf [SUSv4]
catanh [SUSv4]	catanhf [SUSv4]	catanhl [SUSv4]	catanl [SUSv4]
cbrt [SUSv4]	cbrtf [SUSv4]	cbrtl [SUSv4]	ccos [SUSv4]
ccosf [SUSv4]	ccosh [SUSv4]	ccoshf [SUSv4]	ccoshl [SUSv4]
ccosl [SUSv4]	ceil [SUSv4]	ceilf [SUSv4]	ceill [SUSv4]
cexp [SUSv4]	cexpf [SUSv4]	cexpl [SUSv4]	cimag [SUSv4]
cimagf [SUSv4]	cimagl [SUSv4]	clog [SUSv4]	clog10 [LSB]
clog10f [LSB]	clog10l [LSB]	clogf [SUSv4]	clogl [SUSv4]
conj [SUSv4]	conjf [SUSv4]	conjl [SUSv4]	copysign [SUSv4]
copysignf [SUSv4]	copysignl [SUSv4]	cos [SUSv4]	cosf [SUSv4]
cosh [SUSv4]	coshf [SUSv4]	coshl [SUSv4]	cosl [SUSv4]
cpow [SUSv4]	cpowf [SUSv4]	cpowl [SUSv4]	cproj [SUSv4]
cprojf [SUSv4]	cprojl [SUSv4]	creal [SUSv4]	crealf [SUSv4]
creall [SUSv4]	csin [SUSv4]	csinf [SUSv4]	csinh [SUSv4]
csinhf [SUSv4]	csinhl [SUSv4]	csinl [SUSv4]	csqrt [SUSv4]
csqrtf [SUSv4]	csqrtl [SUSv4]	ctan [SUSv4]	ctanf [SUSv4]
ctanh [SUSv4]	ctanhf [SUSv4]	ctanhl [SUSv4]	ctanl [SUSv4]
drem [LSB]	dremf [LSB]	dreml [LSB]	erf [SUSv4]
erfc [SUSv4]	erfcf [SUSv4]	erfcl [SUSv4]	erff [SUSv4]
erfl [SUSv4]	exp [SUSv4]	exp10 [LSB]	exp10f [LSB]
exp10l [LSB]	exp2 [SUSv4]	exp2f [SUSv4]	expf [SUSv4]
expl [SUSv4]	expm1 [SUSv4]	expm1f [SUSv4]	expm1l [SUSv4]

fabs [SUSv4]	fabsf [SUSv4]	fabsl [SUSv4]	fdim [SUSv4]
fdimf [SUSv4]	fdiml [SUSv4]	feclearexcept [SUSv4]	fedisableexcept [LSB]
feenableexcept [LSB]	fegetenv [SUSv4]	fegetexcept [LSB]	fegetexceptflag [SUSv4]
fegetround [SUSv4]	feholdexcept [SUSv4]	feraiseexcept [SUSv4]	fesetenv [SUSv4]
fesetexceptflag [SUSv4]	fesetround [SUSv4]	fetestexcept [SUSv4]	feupdateenv [SUSv4]
finite [LSB]	finitef [LSB]	finitel [LSB]	floor [SUSv4]
floorf [SUSv4]	floorl [SUSv4]	fma [SUSv4]	fmaf [SUSv4]
fmal [SUSv4]	fmax [SUSv4]	fmaxf [SUSv4]	fmaxl [SUSv4]
fmin [SUSv4]	fminf [SUSv4]	fminl [SUSv4]	fmod [SUSv4]
fmodf [SUSv4]	fmodl [SUSv4]	frexp [SUSv4]	frexpf [SUSv4]
frexpl [SUSv4]	gamma [LSB]	gammaf [LSB]	gammal [LSB]
hypot [SUSv4]	hypotf [SUSv4]	hypotl [SUSv4]	ilogb [SUSv4]
ilogbf [SUSv4]	ilogbl [SUSv4]	j0 [SUSv4]	j0f [LSB]
j0l [LSB]	j1 [SUSv4]	j1f [LSB]	j1l [LSB]
jn [SUSv4]	jnf [LSB]	jnl [LSB]	ldexp [SUSv4]
ldexpf [SUSv4]	ldexpl [SUSv4]	lgamma [SUSv4]	lgamma_r [LSB]
lgammaf [SUSv4]	lgammaf_r [LSB]	lgammal [SUSv4]	lgammal_r [LSB]
llrint [SUSv4]	llrintf [SUSv4]	llrintl [SUSv4]	llround [SUSv4]
llroundf [SUSv4]	llroundl [SUSv4]	log [SUSv4]	log10 [SUSv4]
log10f [SUSv4]	log10l [SUSv4]	log1p [SUSv4]	log1pf [SUSv4]
log1pl [SUSv4]	log2 [SUSv4]	log2f [SUSv4]	log2l [SUSv4]
logb [SUSv4]	logbf [SUSv4]	logbl [SUSv4]	logf [SUSv4]
logl [SUSv4]	lrint [SUSv4]	lrintf [SUSv4]	lrintl [SUSv4]
lround [SUSv4]	lroundf [SUSv4]	lroundl [SUSv4]	matherr [LSB]
modf [SUSv4]	modff [SUSv4]	modfl [SUSv4]	nan [SUSv4]
nanf [SUSv4]	nanl [SUSv4]	nearbyint [SUSv4]	nearbyintf [SUSv4]
nearbyintl [SUSv4]	nextafter [SUSv4]	nextafterf [SUSv4]	nextafterl [SUSv4]
nexttoward [SUSv4]	nexttowardf [SUSv4]	nexttowardl [SUSv4]	pow [SUSv4]

pow10 [LSB]	pow10f [LSB]	pow10l [LSB]	powf [SUSv4]
powl [SUSv4]	remainder [SUSv4]	remainderf [SUSv4]	remainderl [SUSv4]
remquo [SUSv4]	remquof [SUSv4]	remquol [SUSv4]	rint [SUSv4]
rintf [SUSv4]	rintl [SUSv4]	round [SUSv4]	roundf [SUSv4]
roundl [SUSv4]	scalb [SUSv3]	scalbf [LSB]	scalbl [LSB]
scalbln [SUSv4]	scalblnf [SUSv4]	scalblnl [SUSv4]	scalbn [SUSv4]
scalbnf [SUSv4]	scalbnl [SUSv4]	significand [LSB]	significandf [LSB]
significandl [LSB]	sin [SUSv4]	sincos [LSB]	sincosf [LSB]
sincosl [LSB]	sinf [SUSv4]	sinh [SUSv4]	sinhf [SUSv4]
sinhl [SUSv4]	sinl [SUSv4]	sqrt [SUSv4]	sqrtf [SUSv4]
sqrtl [SUSv4]	tan [SUSv4]	tanf [SUSv4]	tanh [SUSv4]
tanhf [SUSv4]	tanhl [SUSv4]	tanl [SUSv4]	tgamma [SUSv4]
tgammaf [SUSv4]	tgammal [SUSv4]	trunc [SUSv4]	truncf [SUSv4]
truncl [SUSv4]	y0 [SUSv4]	y0f [LSB]	y0l [LSB]
y1 [SUSv4]	y1f [LSB]	y1l [LSB]	yn [SUSv4]
ynf [LSB]	ynl [LSB]		

An LSB conforming implementation shall provide the generic deprecated functions for Math specified in Table 14-40, 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 14-40 libm - Math Deprecated Function Interfaces

drem [LSB]	dremf [LSB]	dreml [LSB]	finite [LSB]
finitef [LSB]	finitel [LSB]	gamma [LSB]	gammaf [LSB]
gammal [LSB]	matherr [LSB]		

An LSB conforming implementation shall provide the generic data interfaces for Math specified in Table 14-41, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-41 libm - Math Data Interfaces

signgam [SUSv4]			

14.7 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.

14.7.1 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 carg(double complex);
extern float cargf(float complex);
extern long double 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 catanhl(long double complex);
extern long double complex catanl(long double complex);
extern double complex ccos(double complex);
extern float complex ccosf(float complex);
extern 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 cexpl(long double complex);
extern double cimag(double complex);
extern float cimagf(float complex);
extern long double cimagl(long double complex);
extern double complex clog(double complex);
extern double complex clog10(double complex);
```

```
extern float complex clog10f(float complex);
extern long double complex clog10l(long double complex);
extern float complex clogf(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 cproj(double complex);
extern float complex cprojf(float complex);
extern long double complex cprojl(long double complex);
extern double creal(double complex);
extern float crealf(float complex);
extern long double creall(long double complex);
extern double complex csin(double complex);
extern float complex csinf(float complex);
extern double complex csinh(double complex);
extern float complex csinhf(float complex);
extern long double complex csinhl(long double complex);
extern long double complex csinl(long double complex);
extern double complex csqrt(double complex);
extern float complex csqrtf(float complex);
extern long double complex csqrtl(long double complex);
extern double complex ctan(double complex);
extern float complex ctanf(float complex);
extern double complex ctanh(double complex);
extern float complex ctanhf(float complex);
extern long double complex ctanhl(long double complex);
extern long double complex ctanl(long double complex);
```

14.7.2 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);
```

14.7.3 math.h

```
#define DOMAIN  1
#define SING    2

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



```
#define isnormal(x)     (fpclassify (x) == FP_NORMAL)   /* 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  0.63661977236758134308
#define M_LN2   0.69314718055994530942
#define M_SQRT1_2      0.70710678118654752440
#define M_PI_4  0.78539816339744830962
#define M_2_SQRTPI     1.12837916709551257390
#define M_SQRT2 1.41421356237309504880
#define M_LOG2E 1.44269504088896340740
#define M_PI_2  1.57079632679489661923
#define M_LN10  2.30258509299404568402
#define M_E     2.71828182845904523536
#define M_PI    3.14159265358979323846
#define INFINITY       HUGE_VALF

#define MATH_ERRNO     1       /* 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 __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 long double acoshl(long double);
extern long double acosl(long double);
extern double asin(double);
extern float asinf(float);
extern double asinh(double);
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 long double exp10l(long double);
extern double exp2(double);
extern float exp2f(float);
extern float expf(float);
extern long double expl(long double);
extern double expm1(double);
extern float expm1f(float);
extern long double expm1l(long double);
extern double fabs(double);
extern float fabsf(float);
extern long double fabsl(long double);
extern double fdim(double, double);
extern float fdimf(float, float);
extern long double fdiml(long double, long double);
extern int finite(double);
extern int finitef(float);
extern int finitel(long double);
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 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 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 long int llround(double);
extern long long int llroundf(float);
extern long long int llroundl(long double);
extern double log(double);
extern double log10(double);
extern float log10f(float);
extern long double log10l(long double);
extern double log1p(double);
extern float log1pf(float);
extern long double log1pl(long double);
extern double log2(double);
extern float log2f(float);
extern long double log2l(long double);
extern double logb(double);
extern float logbf(float);
extern long double logbl(long double);
extern float logf(float);
extern long double logl(long double);
extern long int lrint(double);
extern long int lrintf(float);
extern long int lrintl(long double);
```

```
extern long int lround(double);
extern long int lroundf(float);
extern long int lroundl(long double);
extern double 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 float powf(float, float);
extern long double powl(long double, long double);
extern double remainder(double, double);
extern float remainderf(float, float);
extern long double remainderl(long double, long double);
extern double remquo(double, double, int *);
extern float remquof(float, float, int *);
extern long double remquol(long double, long double, int *);
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(long double);
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 scalblnf(float, long int);
extern long double scalblnl(long double, long int);
extern double scalbn(double, int);
extern float scalbnf(float, int);
extern long double scalbnl(long double, int);
extern int signgam;
extern double significand(double);
extern float significandf(float);
extern long double significandl(long double);
extern double sin(double);
extern void sincos(double, double *, double *);
extern void sincosf(float, float *, float *);
extern void sincosl(long double, long double *, long double *);
extern float sinf(float);
extern double sinh(double);
extern float sinhf(float);
extern long double sinhl(long double);
extern long double sinl(long double);
extern double sqrt(double);
extern float sqrtf(float);
extern long double sqrtl(long double);
extern double tan(double);
extern float tanf(float);
extern double tanh(double);
extern float tanhf(float);
extern long double tanhl(long 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 truncl(long double);
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 ynl(int, long double);
```

14.8 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 14.6 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009) except that the argument type for __finitef() is known to be float.

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

__finitel

Name

__finitel — test for infinity

Synopsis

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

Description

__finitel() has the same specification as isfinite() in the POSIX 1003.1-2008 (ISO/IEC 9945-2009), except that the argument type for __finitel() is known to be long double.

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

__fpclassify

Name

__fpclassify — Classify real floating type

Synopsis

```
int __fpclassify(double arg);
```

Description

__fpclassify() has the same specification as fpclassify() in POSIX 1003.1-2008 (ISO/IEC 9945-2009), except that the argument type for __fpclassify() is known to be double.

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

__fpclassifyf

Name

__fpclassifyf — Classify real floating type

Synopsis

```
int __fpclassifyf(float arg);
```

Description

__fpclassifyf() has the same specification as fpclassify() in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), except that the argument type for __signbitf() is known to be float.

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

clog10

Name

`clog10` — Logarithm of a Complex Number

Synopsis

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

Description

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

Return Value

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

clog10f

Name

`clog10f` — Logarithm of a Complex Number

Synopsis

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

Description

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

Return Value

The `clog10f()` function shall return the base 10 logarithm.

clog10l

Name

`clog10l` — Logarithm of a Complex Number

Synopsis

```
#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* 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* REM *y* as required by IEC 60559/IEEE 754 Floating Point in the same way as `remainderf()`.

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

Returns

See `remainderf()`.

See Also

`remainderf()`, `drem()`, `dreml()`

dreml

Name

`dreml` — Floating Point Remainder (DEPRECATED)

Synopsis

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

Description

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

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

Returns

See `remainderl()`.

See Also

`remainderl()`, `drem()`, `dremf()`

exp10

Name

`exp10` — Base-10 power function

Synopsis

```
#include <math.h>
double exp10(double x);
```

Description

The `exp10()` function shall return 10^x.

Note: This function is identical to `pow10()`.

Returns

Upon successful completion, `exp10()` shall return 10 rised to the power of `x`.

If the correct value would cause overflow, a range error shall occur and `exp10()` shall return ±HUGE_VAL, with the same sign as the correct value of the function.

See Also

`pow10()`, `exp10f()`, `exp10l()`

exp10f

Name

`exp10f` — Base-10 power function

Synopsis

```
#include <math.h>
float exp10f(float x);
```

Description

The `exp10f()` function shall return 10^x.

Note: This function is identical to `pow10f()`.

Returns

Upon successful completion, `exp10f()` shall return 10 rised to the power of `x`.

If the correct value would cause overflow, a range error shall occur and `exp10f()` shall return ±HUGE_VALF, with the same sign as the correct value of the function.

See Also

`pow10f()`, `exp10()`, `exp10l()`

exp10l

Name

`exp10l` — Base-10 power function

Synopsis

```
#include <math.h>
long double exp10l(long double x);
```

Description

The `exp10l()` function shall return 10^x.

Note: This function is identical to `pow10l()`.

Returns

Upon successful completion, `exp10l()` shall return 10 rised to the power of `x`.

If the correct value would cause overflow, a range error shall occur and `exp10l()` shall return ±HUGE_VALL, with the same sign as the correct value of the function.

See Also

`pow10l()`, `exp10()`, `exp10f()`

fedisableexcept

Name

`fedisableexcept` — disable floating point exceptions

Synopsis

```
#include <fenv.h>
int fedisableexcept(int excepts);
```

Description

The `fedisableexcept()` function disables traps for each of the exceptions represented by the mask `excepts`.

Return Value

The `fedisableexcept()` function returns the previous set of enabled exceptions on success. On error, -1 is returned.

Errors

No errors are defined, but the function will fail if not supported on the architecture.

feenableexcept

Name

`feenableexcept` — enable floating point exceptions

Synopsis

```
#include <fenv.h>
int feenableexcept(int excepts);
```

Description

The `feenableexcept()` function enables traps for each of the exceptions represented by the mask `excepts`.

Return Value

The `feenableexcept()` function returns the previous set of enabled exceptions on success. On error, -1 is returned.

Errors

No errors are defined, but the function will fail if not supported on the architecture.

fegetexcept

Name

`fegetexcept` — query floating point exception handling state

Synopsis

```
#include <fenv.h>
int fegetexcept(void);
```

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
#include <math.h>
int finitel(long double arg);
```

Description

The `finitel()` function shall test whether its argument is neither `INFINITY` nor not a number (NaN).

Returns

On success, `finitel()` shall return 1. Otherwise the function shall return 0.

> **Note:** The ISO C (1999) standard defines the function `isfinite()`, which is more general purpose. The `finitel()` function is deprecated, and applications should use `isfinite()` instead. A future revision of this standard may remove this function.

See Also

`isfinite()`, `finite()`, `finitef()`

gamma

Name

`gamma` — log gamma function (DEPRECATED)

Synopsis

```
#include <math.h>
double gammaf(double x);
```

Description

The `gamma()` function is identical to `lgamma()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

Note: The name `gamma()` for this function is deprecated and should not be used.

Returns

See `lgamma()`.

See Also

`lgamma()`, `lgammaf()`, `lgammal()`, `gammaf()`, `gammal()`

gammaf

Name

`gammaf` — log gamma function (DEPRECATED)

Synopsis

```
#include <math.h>
float gammaf(float x);
```

Description

The `gammaf()` function is identical to `lgammaf()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

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 POSIX 1003.1-2008 (ISO/IEC 9945-2009).

Note: The name `gammal()` for this function is deprecated and should not be used.

Returns

See `lgammal()`.

See Also

`lgamma()`, `lgammaf()`, `lgammal()`, `gamma()`, `gammaf()`

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()`, `ynl()`

j0l

Name

`j0l` — Bessel functions

Synopsis

```
#include <math.h>
long double j0l(long double x);
```

Description

The `j0l()` function is identical to `j0()`, except that the argument *x* and the return value is a long double.

Returns

See `j0()`.

See Also

`j0()`, `j0f()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

j1f

Name

`j1f` — Bessel functions

Synopsis

```
#include <math.h>
float j1f(float x);
```

Description

The `j1f()` function is identical to `j1()`, except that the argument *x* and the return value is a float.

Returns

See `j1()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1l()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

j1l

Name

`j1l` — Bessel functions

Synopsis

```
#include <math.h>
long double j1l(long double x);
```

Description

The `j1l()` function is identical to `j1()`, except that the argument *x* and the return value is a long double.

Returns

See `j0()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

jnf

Name

`jnf` — Bessel functions

Synopsis

```
#include <math.h>
float jnf(float x);
```

Description

The `jnf()` function is identical to `jn()`, except that the argument *x* and the return value is a float.

Returns

See `jn()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnl()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

jnl

Name

`jnl` — Bessel functions

Synopsis

```
#include <math.h>
long double jnl(long double x);
```

Description

The `jnl()` function is identical to `jn()`, except that the argument *x* and the return value is a long double.

Returns

See `jn()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnf()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

lgamma_r

Name

`lgamma_r` — log gamma functions

Synopsis

```
#include <math.h>
double lgamma_r(double x, int * signp);
```

Description

The `lgamma_r()` function shall compute the natural logarithm of the absolute value of the Gamma function, as `lgamma()`. However, instead of setting the external integer `signgam` to the sign of the Gamma function, `lgamma_r()` shall set the integer referenced by *signp* to the sign.

Returns

See `lgamma()` and `signgam`.

See Also

`lgamma()`, `lgammaf_r()`, `lgammal_r()`, `signgam`

lgammaf_r

Name

`lgammaf_r` — log gamma functions

Synopsis

```
#include <math.h>
float lgammaf_r(float x, int * signp);
```

Description

The `lgammaf_r()` function shall compute the natural logarithm of the absolute value of the Gamma function, as `lgammaf()`. However, instead of setting the external integer `signgam` to the sign of the Gamma function, `lgammaf_r()` shall set the integer referenced by *`signp`* to the sign.

Returns

See `lgammaf()` and `signgam`.

See Also

`lgamma()`, `lgamma_r()`, `lgammal_r()`, `signgam`

lgammal_r

Name

`lgammal_r` — log gamma functions

Synopsis

```
#include <math.h>
double lgammal_r(double x, int * signp);
```

Description

The `lgammal_r()` function shall compute the natural logarithm of the absolute value of the Gamma function, as `lgammal()`. However, instead of setting the external integer `signgam` to the sign of the Gamma function, `lgammal_r()` shall set the integer referenced by *`signp`* to the sign.

Returns

See `lgammal()` and `signgam`.

See Also

`lgamma()`, `lgamma_r()`, `lgammaf_r()`, `signgam`

matherr

Name

`matherr` — math library exception handling

Synopsis

```
#include <math.h>
int matherr(struct exception *__exc);
```

Description

The System V Interface Definition (SVID) Issue 3 specifies that various math functions should invoke a function called `matherr()` if a math exception is detected. This function is called before the math function returns; after `matherr()` returns, the system then returns to the math function, which in turn returns to the caller.

`matherr()` is obsolete; indeed it was withdrawn in the System V Interface Definition (SVID) Issue 4, and is required only by this specification for historical compatibility, and will be removed in a future version. The floating point environment function group including `fesetenv()` should be used instead.

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

See Also

`fesetenv()`, `fegetenv()`, `feupdateenv()`.

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* ≡ `sig` × 2^n scaled such that 1 ≤ `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 ≤ `sig` < 2.

If *x* is 0, ±HUGE_VAL, or NaN, the result is undefined.

See Also

`significandf()`, `significandl()`

significandf

Name

`significandf` — floating point mantissa

Synopsis

```
#include <math.h>
float significandf(float x);
```

Description

The `significandf()` function shall return the mantissa of *x*, `sig` such that *x* ≡ `sig` × 2^n scaled such that 1 ≤ `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 ≤ `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

```
#include <math.h>
long double significandl(long double x);
```

Description

The `significandl()` function shall return the mantissa of *x*, `sig` such that *x* ≡ `sig` × 2^n scaled such that 1 ≤ `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 ≤ `sig` < 2.

If *x* is 0, ±HUGE_VALL, or NaN, the result is undefined.

See Also

`significand()`, `significandf()`

sincos

Name

`sincos` — trigonometric functions

Synopsis

```
#define _GNU_SOURCE
#include <math.h>
void sincos(double x, double * sin, double * cos);
```

Description

The `sincos()` function shall calculate both the sine and cosine of *x*. The sine shall be stored in the location referenced by *sin*, and the cosine in the location referenced by *cosine*.

Returns

None. See `sin()` and `cos()` for possible error conditions.

See Also

`cos()`, `sin()`, `sincosf()`, `sincosl()`

sincosf

Name

`sincosf` — trigonometric functions

Synopsis

```
#define _GNU_SOURCE
#include <math.h>
void sincosf(float x, float * sin, float * cos);
```

Description

The `sincosf()` function shall calculate both the sine and cosine of *x*. The sine shall be stored in the location referenced by *sin*, and the cosine in the location referenced by *cosine*.

Returns

None. See `sin()` and `cos()` for possible error conditions.

See Also

`cos()`, `sin()`, `sincos()`, `sincosl()`

sincosl

Name

`sincosl` — trigonometric functions

Synopsis

```
#define _GNU_SOURCE
#include <math.h>
void sincosl(long double x, long double * sin, long double * cos);
```

Description

The `sincosl()` function shall calculate both the sine and cosine of *x*. The sine shall be stored in the location referenced by *sin*, and the cosine in the location referenced by *cosine*.

Returns

None. See `sin()` and `cos()` for possible error conditions.

See Also

`cos()`, `sin()`, `sincos()`, `sincosl()`

y0f

Name

`y0f` — Bessel functions

Synopsis

```
#include <math.h>
float y0f(float x);
```

Description

The `y0f()` function is identical to `y0()`, except that the argument *x* and the return value is a float.

Returns

See `y0()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0l()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

y0l

Name

`y0l` — Bessel functions

Synopsis

```
#include <math.h>
long double y0l(long double x);
```

Description

The `y0l()` function is identical to `y0()`, except that the argument *x* and the return value is a long double.

Returns

See `y0()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0f()`, `y1()`, `y1f()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

y1f

Name

`y1f` — Bessel functions

Synopsis

```
#include <math.h>
float y1f(float x);
```

Description

The `y1f()` function is identical to `y1()`, except that the argument *x* and the return value is a float.

Returns

See `y1()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1l()`, `yn()`, `ynf()`, `ynl()`

y1l

Name

`y1l` — Bessel functions

Synopsis

```
#include <math.h>
long double y1l(long double x);
```

Description

The `y1l()` function is identical to `y1()`, except that the argument *x* and the return value is a long double.

Returns

See `j0()`.

See Also

`j0()`, `j0f()`, `j0l()`, `j1()`, `j1f()`, `j1l()`, `jn()`, `jnf()`, `jnl()`, `y0()`, `y0f()`, `y0l()`, `y1()`, `y1f()`, `yn()`, `ynf()`, `ynl()`

ynf

Name

ynf — Bessel functions

Synopsis

```
#include <math.h>
float ynf(float x);
```

Description

The ynf() function is identical to yn(), except that the argument *x* and the return value is a float.

Returns

See yn().

See Also

j0(),j0f(),j0l(),j1(),j1f(),j1l(),jn(),jnf(),jnl(),y0(),y0f(),y0l(), y1(),y1f(),y1l(),yn(),ynl()

ynl

Name

ynl — Bessel functions

Synopsis

```
#include <math.h>
long double ynl(long double x);
```

Description

The ynl() function is identical to yn(), except that the argument *x* and the return value is a long double.

Returns

See yn().

See Also

j0(),j0f(),j0l(),j1(),j1f(),j1l(),jn(),jnf(),jnl(),y0(),y0f(),y0l(), y1(),y1f(),y1l(),yn(),ynf()

14.9 Interfaces for libpthread

Table 14-42 defines the library name and shared object name for the libpthread library

Table 14-42 libpthread Definition

Library:	libpthread
SONAME:	libpthread.so.0

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

[LFS] Large File Support
[LSB] This Specification
[SUSv3] POSIX 1003.1-2001 (ISO/IEC 9945-2003)
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

14.9.1 Realtime Threads

14.9.1.1 Interfaces for Realtime Threads

An LSB conforming implementation shall provide the generic functions for Realtime Threads specified in Table 14-43, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-43 libpthread - Realtime Threads Function Interfaces

pthread_attr_getinheritsched [SUSv4]	pthread_attr_getschedpolicy [SUSv4]	pthread_attr_getscope [SUSv4]	pthread_attr_setinheritsched [SUSv4]
pthread_attr_setschedpolicy [SUSv4]	pthread_attr_setscope [SUSv4]	pthread_getschedparam [SUSv4]	pthread_mutex_getprioceiling(GLIBC_2.4) [SUSv4]
pthread_mutex_setprioceiling(GLIBC_2.4) [SUSv4]	pthread_mutexattr_getprioceiling(GLIBC_2.4) [SUSv4]	pthread_mutexattr_getprotocol(GLIBC_2.4) [SUSv4]	pthread_mutexattr_setprioceiling(GLIBC_2.4) [SUSv4]
pthread_mutexattr_setprotocol(GLIBC_2.4) [SUSv4]	pthread_setschedparam [SUSv4]	pthread_setschedprio(GLIBC_2.3.4) [SUSv4]	

14.9.2 Advanced Realtime Threads

14.9.2.1 Interfaces for Advanced Realtime Threads

An LSB conforming implementation shall provide the generic functions for Advanced Realtime Threads specified in Table 14-44, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-44 libpthread - Advanced Realtime Threads Function Interfaces

pthread_barrier_destroy [SUSv4]	pthread_barrier_init [SUSv4]	pthread_barrier_wait [SUSv4]	pthread_barrierattr_destroy [SUSv4]
pthread_barrierattr_getpshared(GLIBC_2.3.3) [SUSv4]	pthread_barrierattr_init [SUSv4]	pthread_barrierattr_setpshared [SUSv4]	pthread_getcpuclockid [SUSv4]

pthread_spin_destroy [SUSv4]	pthread_spin_init [SUSv4]	pthread_spin_lock [SUSv4]	pthread_spin_trylock [SUSv4]
pthread_spin_unlock [SUSv4]			

14.9.3 Posix Threads

14.9.3.1 Interfaces for Posix Threads

An LSB conforming implementation shall provide the generic functions for Posix Threads specified in Table 14-45, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-45 libpthread - Posix Threads Function Interfaces

_pthread_cleanup_pop [LSB]	_pthread_cleanup_push [LSB]	pthread_attr_destroy [SUSv4]	pthread_attr_getdetachstate [SUSv4]
pthread_attr_getguardsize [SUSv4]	pthread_attr_getschedparam [SUSv4]	pthread_attr_getstack [SUSv4]	pthread_attr_getstackaddr [SUSv3]
pthread_attr_getstacksize [SUSv4]	pthread_attr_init [SUSv4]	pthread_attr_setdetachstate [SUSv4]	pthread_attr_setguardsize [SUSv4]
pthread_attr_setschedparam [SUSv4]	pthread_attr_setstack [SUSv4]	pthread_attr_setstackaddr [SUSv3]	pthread_attr_setstacksize [SUSv4]
pthread_cancel [SUSv4]	pthread_cond_broadcast [SUSv4]	pthread_cond_destroy [SUSv4]	pthread_cond_init [SUSv4]
pthread_cond_signal [SUSv4]	pthread_cond_timedwait [SUSv4]	pthread_cond_wait [SUSv4]	pthread_condattr_destroy [SUSv4]
pthread_condattr_getclock(GLIBC_2.3.3) [SUSv4]	pthread_condattr_getpshared [SUSv4]	pthread_condattr_init [SUSv4]	pthread_condattr_setclock(GLIBC_2.3.3) [SUSv4]
pthread_condattr_setpshared [SUSv4]	pthread_create [SUSv4]	pthread_detach [SUSv4]	pthread_equal [SUSv4]
pthread_exit [SUSv4]	pthread_getconcurrency [SUSv4]	pthread_getspecific [SUSv4]	pthread_join [SUSv4]
pthread_key_create [SUSv4]	pthread_key_delete [SUSv4]	pthread_kill [SUSv4]	pthread_mutex_consistent(GLIBC_2.12) [SUSv4]
pthread_mutex_destroy [SUSv4]	pthread_mutex_init [SUSv4]	pthread_mutex_lock [SUSv4]	pthread_mutex_timedlock [SUSv4]

pthread_mutex_trylock [SUSv4]	pthread_mutex_unlock [SUSv4]	pthread_mutexattr_destroy [SUSv4]	pthread_mutexattr_getpshared [SUSv4]
pthread_mutexattr_getrobust(GLIBC_2.12) [SUSv4]	pthread_mutexattr_gettype [SUSv4]	pthread_mutexattr_init [SUSv4]	pthread_mutexattr_setpshared [SUSv4]
pthread_mutexattr_setrobust(GLIBC_2.12) [SUSv4]	pthread_mutexattr_settype [SUSv4]	pthread_once [SUSv4]	pthread_rwlock_destroy [SUSv4]
pthread_rwlock_init [SUSv4]	pthread_rwlock_rdlock [SUSv4]	pthread_rwlock_timedrdlock [SUSv4]	pthread_rwlock_timedwrlock [SUSv4]
pthread_rwlock_tryrdlock [SUSv4]	pthread_rwlock_trywrlock [SUSv4]	pthread_rwlock_unlock [SUSv4]	pthread_rwlock_wrlock [SUSv4]
pthread_rwlockattr_destroy [SUSv4]	pthread_rwlockattr_getpshared [SUSv4]	pthread_rwlockattr_init [SUSv4]	pthread_rwlockattr_setpshared [SUSv4]
pthread_self [SUSv4]	pthread_setcancelstate [SUSv4]	pthread_setcanceltype [SUSv4]	pthread_setconcurrency [SUSv4]
pthread_setspecific [SUSv4]	pthread_sigmask [SUSv4]	pthread_testcancel [SUSv4]	sem_close [SUSv4]
sem_destroy [SUSv4]	sem_getvalue [SUSv4]	sem_init [SUSv4]	sem_open [SUSv4]
sem_post [SUSv4]	sem_timedwait [SUSv4]	sem_trywait [SUSv4]	sem_unlink [SUSv4]
sem_wait [SUSv4]			

An LSB conforming implementation shall provide the generic deprecated functions for Posix Threads specified in Table 14-46, 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 14-46 libpthread - Posix Threads Deprecated Function Interfaces

pthread_attr_getstackaddr [SUSv3]	pthread_attr_setstackaddr [SUSv3]		

14.9.4 Thread aware versions of libc interfaces

14.9.4.1 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 14-47, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-47 libpthread - Thread aware versions of libc interfaces Function Interfaces

lseek64 [LFS]	open64 [LFS]	pread [SUSv4]	pread64 [LSB]
pwrite [SUSv4]	pwrite64 [LSB]		

14.9.5 GNU Extensions for libpthread

14.9.5.1 Interfaces for GNU Extensions for libpthread

An LSB conforming implementation shall provide the generic functions for GNU Extensions for libpthread specified in Table 14-48, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-48 libpthread - GNU Extensions for libpthread Function Interfaces

pthread_getattr_np [LSB]	pthread_mutex_consistent_np [LSB]	pthread_mutexattr_getrobust_np [LSB]	pthread_mutexattr_setrobust_np [LSB]
pthread_rwlockattr_getkind_np [LSB]	pthread_rwlockattr_setkind_np [LSB]		

14.9.6 System Calls

14.9.6.1 Interfaces for System Calls

An LSB conforming implementation shall provide the generic functions for System Calls specified in Table 14-49, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-49 libpthread - System Calls Function Interfaces

close [SUSv4]	fcntl [LSB]	fork [SUSv4]	fsync [SUSv4]
lseek [SUSv4]	msync [SUSv4]	nanosleep [SUSv4]	open [SUSv4]
pause [SUSv4]	read [SUSv4]	vfork [SUSv3]	wait [SUSv4]
waitpid [LSB]	write [SUSv4]		

14.9.7 Standard I/O

14.9.7.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the generic functions for Standard I/O specified in Table 14-50, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-50 libpthread - Standard I/O Function Interfaces

flockfile [SUSv4]			

14.9.8 Signal Handling

14.9.8.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the generic functions for Signal Handling specified in Table 14-51, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-51 libpthread - Signal Handling Function Interfaces

__libc_current_sigrtmax [LSB]	__libc_current_sigrtmin [LSB]	raise [SUSv4]	sigaction [SUSv4]
siglongjmp [SUSv4]	sigwait [SUSv4]		

14.9.9 Standard Library

14.9.9.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the generic functions for Standard Library specified in Table 14-52, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-52 libpthread - Standard Library Function Interfaces

__errno_location [LSB]	ftrylockfile [SUSv4]	funlockfile [SUSv4]	longjmp [SUSv4]
system [LSB]			

14.9.10 Socket Interface

14.9.10.1 Interfaces for Socket Interface

An LSB conforming implementation shall provide the generic functions for Socket Interface specified in Table 14-53, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-53 libpthread - Socket Interface Function Interfaces

__h_errno_location [LSB]	accept [SUSv4]	connect [SUSv4]	recv [SUSv4]

recvfrom [SUSv4]	recvmsg [SUSv4]	send [SUSv4]	sendmsg [SUSv4]
sendto [SUSv4]			

14.9.11 Terminal Interface Functions

14.9.11.1 Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the generic functions for Terminal Interface Functions specified in Table 14-54, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-54 libpthread - Terminal Interface Functions Function Interfaces

tcdrain [SUSv4]			

14.10 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.

14.10.1 lsb/pthread.h

```
typedef unsigned long int pthread_t;
```

14.10.2 pthread.h

```
#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 __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;
enum {
    PTHREAD_PRIO_NONE,
    PTHREAD_PRIO_INHERIT,
    PTHREAD_PRIO_PROTECT
};
enum {
    PTHREAD_MUTEX_STALLED = 0,
    PTHREAD_MUTEX_STALLED_NP = 0,
    PTHREAD_MUTEX_ROBUST = 1,
    PTHREAD_MUTEX_ROBUST_NP = 1
};
enum {
    PTHREAD_RWLOCK_PREFER_READER_NP,
    PTHREAD_RWLOCK_PREFER_WRITER_NP,
    PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP,
    PTHREAD_RWLOCK_DEFAULT_NP = PTHREAD_RWLOCK_PREFER_READER_NP
};

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_atfork(void (*__prepare) (void),
                          void  (*__parent)  (void),  void  (*__child)
(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_setschedparam(pthread_attr_t * __attr,
                                      const struct sched_param *__param);
extern int pthread_attr_setschedpolicy(pthread_attr_t * __attr,
                                       int __policy);
```



```
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_getclock(const pthread_condattr_t *
attr,
                                     clockid_t * clock_id);
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_setclock(pthread_condattr_t * attr,
                                     clockid_t clock_id);
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_getattr_np(pthread_t thread, pthread_attr_t *
attr);
extern int pthread_getconcurrency(void);
extern int pthread_getcpuclockid(pthread_t __thread_id,
                                 clockid_t * __clock_id);
extern int pthread_getschedparam(pthread_t __target_thread, int
*__policy,
                                 struct sched_param *__param);
extern void *pthread_getspecific(pthread_key_t __key);
extern int pthread_join(pthread_t __th, void **__thread_return);
extern int pthread_key_create(pthread_key_t * __key,
                              void (*__destr_function) (void *));
extern int pthread_key_delete(pthread_key_t __key);
```

```
extern int pthread_mutex_consistent(pthread_mutex_t * mutex);
extern int pthread_mutex_consistent_np(pthread_mutex_t * __mutex);
extern int pthread_mutex_destroy(pthread_mutex_t * __mutex);
extern int pthread_mutex_getprioceiling(const pthread_mutex_t *
__mutex,
                                        int *__prioceiling);
extern int pthread_mutex_init(pthread_mutex_t * __mutex,
                              const pthread_mutexattr_t * __mutexattr);
extern int pthread_mutex_lock(pthread_mutex_t * __mutex);
extern int pthread_mutex_setprioceiling(pthread_mutex_t * __mutex,
                                        int __prioceiling,
                                        int *__old_ceiling);
extern int pthread_mutex_timedlock(pthread_mutex_t * __mutex,
                                   const struct timespec *__abstime);
extern int pthread_mutex_trylock(pthread_mutex_t * __mutex);
extern int pthread_mutex_unlock(pthread_mutex_t * __mutex);
extern int pthread_mutexattr_destroy(pthread_mutexattr_t * __attr);
extern int pthread_mutexattr_getprioceiling(const
pthread_mutexattr_t *
                                            __attr, int *__prioceiling);
extern int pthread_mutexattr_getprotocol(const pthread_mutexattr_t
*
                                         __attr, int *__protocol);
extern int pthread_mutexattr_getpshared(const pthread_mutexattr_t
* __attr,
                                        int *__pshared);
extern int pthread_mutexattr_getrobust(const pthread_mutexattr_t *
attr,
                                       int *robust);
extern int pthread_mutexattr_getrobust_np(const
pthread_mutexattr_t *
                                          __attr, int *__robustness);
extern int pthread_mutexattr_gettype(const pthread_mutexattr_t *
__attr,
                                     int *__kind);
extern int pthread_mutexattr_init(pthread_mutexattr_t * __attr);
extern int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *
__attr,
                                            int __prioceiling);
extern int pthread_mutexattr_setprotocol(pthread_mutexattr_t *
__attr,
                                         int __protocol);
extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *
__attr,
                                        int __pshared);
extern int pthread_mutexattr_setrobust(pthread_mutexattr_t * attr,
                                       int robust);
extern int pthread_mutexattr_setrobust_np(pthread_mutexattr_t *
__attr,
                                          int __robustness);
extern int pthread_mutexattr_settype(pthread_mutexattr_t * __attr,
                                     int __kind);
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_rwlock_unlock(pthread_rwlock_t * __rwlock);
extern int pthread_rwlock_wrlock(pthread_rwlock_t * __rwlock);
```

```
extern   int   pthread_rwlockattr_destroy(pthread_rwlockattr_t   *
__attr);
extern          int          pthread_rwlockattr_getkind_np(const
pthread_rwlockattr_t *
                                    __attr, int *__pref);
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_setkind_np(pthread_rwlockattr_t  *
__attr,
                                    int __pref);
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);
```

14.10.3 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
__value);
extern sem_t *sem_open(const char *__name, int __oflag, ...);
extern int sem_post(sem_t * __sem);
extern  int  sem_timedwait(sem_t  *  __sem,  const  struct  timespec
*__abstime);
extern int sem_trywait(sem_t * __sem);
extern int sem_unlink(const char *__name);
extern int sem_wait(sem_t * __sem);
```

14.11 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 14.9 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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*.

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 *POSIX 1003.1-2008 (ISO/IEC 9945-2009)*.

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

pthread_getattr_np

Name

`pthread_getattr_np` — get thread attributes

Synopsis

```
#include <pthread.h>
int pthread_getattr_np(pthread_t thread, pthread_attr_t *attr);
```

Description

`pthread_getattr_np()` fills in the thread attribute object *attr* with attribute values describing the running thread *thread*. This is useful to detect runtime changes from the values specified in the thread attributes object used to create the thread with `pthread_create()`. The following differences may be noted:

- The detach state, since a joinable thread may have detached itself after creation. Use `pthread_attr_getdetachstate()` to extract from *attr*.
- The stack size, which the implementation may align to a suitable boundary. Use `pthread_attr_getstack()` to extract from *attr*.
- The guard size, which the implementation may round upwards to a multiple of the page size, or ignore (i.e., treat as 0), if the application is allocating its own stack. Use `pthread_attr_getguardsize()` to extract from *attr*.

If the stack address attribute was not set in the thread attributes object used to create the thread, then the thread attributes object returned by `pthread_getattr_np()` will show the actual stack address the implementation selected for the thread. Use `pthread_attr_getstack()` to extract from *attr*.

The thread attributes object *attr* should be destroyed using `pthread_attr_destroy()` when it is no longer needed.

Return Value

On success, `pthread_getattr_np()` returns 0; on error, it returns a non-zero error number.

Errors

ENOMEM

Insufficient memory to complete the operation.

In addition, if *thread* refers to the main thread, then `pthread_getattr_np()` may also fail due to errors from various underlying calls: `fopen()`, if the pseudo-file containing the memory region map cannot be opened; `getrlimit()` if the `RLIMIT_STACK` resource limit it not supported.

Notes

This function is a `GNU` extension.

See Also

pthread_attr_destroy(), pthread_attr_getdetachstate(), pthread_attr_getguardsize(), pthread_attr_getstack(), pthread_create().

pthread_mutex_consistent_np

Name

pthread_mutex_consistent_np — mark state protected by robust mutex as consistent

Synopsis

```
#include <pthread.h>
int pthread_mutex_consistent_np(pthread_mutex_t * __mutex);
```

Description

pthread_mutex_consistent_np() shall behave as described for pthread_mutex_consistent() in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

pthread_mutexattr_getrobust_np, pthread_mutexattr_setrobust_np

Name

pthread_mutexattr_getrobust_np, pthread_mutexattr_setrobust_np — get and set the mutex robust attribute

Synopsis

```
#include <pthread.h>
int pthread_mutexattr_getrobust_np(const pthread_mutexattr_t * __attr,
int * __robustness);
int pthread_mutexattr_setrobust_np(const pthread_mutexattr_t * __attr,
int __robustness);
```

Description

pthread_mutexattr_setrobust_np() shall behave as described for pthread_mutexattr_setrobust() in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

pthread_mutexattr_getrobust_np() shall behave as described for pthread_mutexattr_getrobust() in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

Two additional valid values are defined for *__robustness*: PTHREAD_MUTEX_STALLED_NP, which is identical to PTHREAD_MUTEX_STALLED and PTHREAD_MUTEX_ROBUST_NP, which is identical to PTHREAD_MUTEX_ROBUST.

pthread_rwlockattr_getkind_np, pthread_rwlockattr_setkind_np

Name

`pthread_rwlockattr_getkind_np`, `pthread_rwlockattr_setkind_np` — get/set the read-write lock kind of the thread read-write lock attribute object

Synopsis

```
#include <pthread.h>
int pthread_rwlockattr_getkind_np(const pthread_rwlockattr_t * attr,
int * pref);
int pthread_rwlockattr_setkind_np(pthread_rwlockattr_t * attr, int *
pref);
```

Description

The `pthread_rwlockattr_setkind_np()` function sets the kind of read-write lock of the thread read-write lock attribute object referred to by attr to the value specified with pref. The argument pref may be set to `PTHREAD_RWLOCK_PREFER_READER_NP`, `PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP`, or `PTHREAD_RWLOCK_PREFER_WRITER_NP`. The default lock setting is `PTHREAD_RWLOCK_PREFER_READER_NP`. A thread may hold multiple read locks, i.e. read locks are recursive. According to The Single Unix Specification, the behavior is unspecified when a reader tries to place a lock, and there is no write lock but writers are waiting. Giving preference to the reader, as is set by default with the `PTHREAD_RWLOCK_PREFER_READER_NP` value implies that the reader will receive the requested lock, even if a writer is waiting. As long as there are readers the writer will be starved. Setting the kind to `PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP`, avoids writer starvation as long as any read locking is not done in a recursive fashion. The `pthread_rwlockattr_getkind_np()` function returns the value of the read-write lock attribute of the thread read-write lock attribute object referred to by *attr* in the pointer *pref*.

Return Value

`pthread_rwlockattr_setkind_np()` function returns 0 on success; on error, it returns a non-zero error number. `pthread_rwlockattr_setkind_np()` function always returns 0.

Errors

EINVAL

pref is set to an unsupported value.

Notes

Setting the value read-write lock kind to PTHREAD_RWLOCK_PREFER_WRITER_NP, results in the same behavior as setting the value to PTHREAD_RWLOCK_PREFER_READER_NP. As long as a reader thread holds the lock the thread holding a write lock will be starved. Setting the kind value to PTHREAD_RWLOCK_PREFER_WRITER_NONRECURSIVE_NP, allows the writer to run. However, the writer may not be recursive as is implied by the name.

waitpid

Name

waitpid — wait for child process

Description

waitpid() is as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with differences as listed below.

Need not support WCONTINUED or WIFCONTINUED

Implementations need not support the XSI optional functionality of WCONTINUED() or WIFCONTINUED().

14.12 Interfaces for libgcc_s

Table 14-55 defines the library name and shared object name for the libgcc_s library

Table 14-55 libgcc_s Definition

Library:	libgcc_s
SONAME:	libgcc_s.so.1

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

[LSB] This Specification

14.12.1 Unwind Library

14.12.1.1 Interfaces for Unwind Library

An LSB conforming implementation shall provide the generic functions for Unwind Library specified in Table 14-56, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-56 libgcc_s - Unwind Library Function Interfaces

_Unwind_Backtrace [LSB]	_Unwind_DeleteException [LSB]	_Unwind_FindEnclosingFunction [LSB]	_Unwind_ForcedUnwind [LSB]
_Unwind_GetCFA [LSB]	_Unwind_GetGR [LSB]	_Unwind_GetIP [LSB]	_Unwind_GetIPInfo(GCC_4.2.0) [LSB]

_Unwind_GetLanguageSpecificData [LSB]	_Unwind_GetRegionStart [LSB]	_Unwind_RaiseException [LSB]	_Unwind_Resume [LSB]
_Unwind_Resume_or_Rethrow [LSB]	_Unwind_SetGR [LSB]	_Unwind_SetIP [LSB]	

14.13 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.

14.13.1 unwind.h

```
struct _Unwind_Context;
struct _Unwind_Exception;

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) (_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;

typedef _Unwind_Reason_Code(*_Unwind_Stop_Fn) (int version,
                                        _Unwind_Action actions,
                                        _Unwind_Exception_Class
                                        exceptionClass,
                                        struct _Unwind_Exception *
                                        exceptionObject,
                                        struct _Unwind_Context *
                                        context,
                                        void *stop_parameter);

typedef       _Unwind_Reason_Code(*_Unwind_Trace_Fn)       (struct
_Unwind_Context *,
                                        void *);
extern   _Unwind_Reason_Code   _Unwind_Backtrace(_Unwind_Trace_Fn,
void *);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern       _Unwind_Reason_Code      _Unwind_ForcedUnwind(struct
_Unwind_Exception *,
                                        _Unwind_Stop_Fn, void *);
extern _Unwind_Word _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetIPInfo(struct _Unwind_Context *, int
*);
extern         void         *_Unwind_GetLanguageSpecificData(struct
_Unwind_Context *);
extern  _Unwind_Ptr _Unwind_GetRegionStart(struct  _Unwind_Context
*);
extern      _Unwind_Reason_Code      _Unwind_RaiseException(struct
_Unwind_Exception
                                        *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
                                        _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
```

14.14 Interface Definitions for libgcc_s

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

Other interfaces listed in Section 14.12 shall behave as described in the referenced base document.

_Unwind_Backtrace

Name

`_Unwind_Backtrace` — private C++ error handling method

Synopsis

```
_Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn trace, void *
trace_argument);
```

Description

`_Unwind_Backtrace()` performs a stack backtrace using unwind data. The *trace* callback is called for every stack frame in the call chain. No cleanup actions are performed.

_Unwind_DeleteException

Name

`_Unwind_DeleteException` — private C++ error handling method

Synopsis

```
void _Unwind_DeleteException(struct _Unwind_Exception * object);
```

Description

`_Unwind_DeleteException()` deletes the given exception *object*. If a given runtime resumes normal execution after catching a foreign exception, it will not know how to delete that exception. Such an exception shall be deleted by calling `_Unwind_DeleteException()`. This is a convenience function that calls the function pointed to by the *exception_cleanup* field of the exception header.

_Unwind_FindEnclosingFunction

Name

`_Unwind_FindEnclosingFunction` — private C++ error handling method

Synopsis

```
void * _Unwind_FindEnclosingFunction(void * ip);
```

Description

`_Unwind_FindEnclosingFunction()` Find the start address of the procedure containing the specified *ip* or NULL if it cannot be found (for example, because the function has no unwind info).

Note that there is not necessarily a one-to-one correspondence between source level functions and procedures. Some functions do not have unwind-info and others are split into multiple procedures.

_Unwind_ForcedUnwind

Name

_Unwind_ForcedUnwind — private C++ error handling method

Synopsis

```
#include <unwind.h>
_Unwind_Reason_Code _Unwind_ForcedUnwind(struct _Unwind_Exception *
object, _Unwind_Stop_Fn stop, void * stop_parameter);
```

Description

Forced unwinding is a single-phase process. *stop* and *stop_parameter* control the termination of the unwind process instead of the usual personality routine query. Stop function *stop* is called for each unwind frame, with the parameteres described for the usual personality routine below, plus an additional *stop_parameter*.

Return Value

When *stop* identifies the destination frame, it transfers control to the user code as appropriate without returning, normally after calling _Unwind_DeleteException(). If not, then it should return an _Unwind_Reason_Code value.

If *stop* returns any reason code other than _URC_NO_REASON, then the stack state is indeterminate from the point of view of the caller of _Unwind_ForcedUnwind(). Rather than attempt to return, therefore, the unwind library should use the *exception_cleanup* entry in *object*, and then call abort().

_URC_NO_REASON

This is not the destination from. The unwind runtime will call frame's personality routine with the _UA_FORCE_UNWIND and _UA_CLEANUP_PHASE flag set in *actions*, and then unwind to the next frame and call the stop() function again.

_URC_END_OF_STACK

In order to allow _Unwind_ForcedUnwind() to perform special processing when it reaches the end of the stack, the unwind runtime will call it after the last frame is rejected, with a NULL stack pointer in the context, and the STOP() FUNCTION SHALL CATCH THIS CONDITION. iT MAY return this code if it cannot handle end-of-stack.

_URC_FATAL_PHASE2_ERROR

The stop() function may return this code for other fatal conditions like stack corruption.

_Unwind_GetCFA

Name

_Unwind_GetCFA — private C++ error handling method

Synopsis

```
_Unwind_Word _Unwind_GetCFA(struct _Unwind_Context * context);
```

Description

_Unwind_GetCFA() shall retrieve the value of the Canonical Frame Address (CFA) of the given *context*.

_Unwind_GetGR

Name

_Unwind_GetGR — private C++ error handling method

Synopsis

```
_Unwind_Word _Unwind_GetGR(struct _Unwind_Context * context, int index);
```

Description

_Unwind_GetGR() returns data at *index* found in *context*. The register is identified by its index: 0 to 31 are for the fixed registers, and 32 to 127 are for the stacked registers.

During the two phases of unwinding, only GR1 has a guaranteed value, which is the global pointer of the frame referenced by the unwind *context*. If the register has its NAT bit set, the behavior is unspecified.

_Unwind_GetIP

Name

_Unwind_GetIP — private C++ error handling method

Synopsis

```
_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);
```

Description

_Unwind_GetIP() returns the instruction pointer value for the routine identified by the unwind *context*.

_Unwind_GetIPInfo

Name

`_Unwind_GetIPInfo` — private C++ error handling method

Synopsis

```
_Unwind_Ptr _Unwind_GetIPInfo(struct _Unwind_Context * context, int *
ip_before_insn);
```

Description

`_Unwind_GetIPInfo()` returns the instruction pointer value for the routine identified by the unwind *`context`* and sets *`ip_before_insn`* flag indicating whether that IP is before or after first not yet fully executed instruction.

_Unwind_GetLanguageSpecificData

Name

`_Unwind_GetLanguageSpecificData` — private C++ error handling method

Synopsis

```
#include <unwind.h>
_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *
context);
```

Description

`_Unwind_GetLanguageSpecificData()` returns the address of the language specific data area for the current stack frame described by *`context`*.

_Unwind_GetRegionStart

Name

`_Unwind_GetRegionStart` — private C++ error handling method

Synopsis

```
_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);
```

Description

`_Unwind_GetRegionStart()` routine returns the address (i.e., `0`) of the beginning of the procedure or code fragment described by the current unwind descriptor block.

_Unwind_RaiseException

Name

`_Unwind_RaiseException` — private C++ error handling method

Synopsis

```
_Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception
* object);
```

Description

`_Unwind_RaiseException()` raises an exception, passing along the given exception *`object`*, which should have its *`exception_class`* and *`exception_cleanup`* fields set. The exception object has been allocated by the language-specific runtime, and has a language-specific format, except that it shall contain an `_Unwind_Exception`.

Return Value

`_Unwind_RaiseException()` does not return unless an error condition is found. If an error condition occurs, an `_Unwind_Reason_Code` is returnd:

_URC_END_OF_STACK

The unwinder encountered the end of the stack during phase one without finding a handler. The unwind runtime will not have modified the stack. The C++ runtime will normally call `uncaught_exception()` in this case.

_URC_FATAL_PHASE1_ERROR

The unwinder encountered an unexpected error during phase one, because of something like stack corruption. The unwind runtime will not have modified the stack. The C++ runtime will normally call `terminate()` in this case.

_URC_FATAL_PHASE2_ERROR

The unwinder encountered an unexpected error during phase two. This is usually a *throw*, which will call `terminate()`.

_Unwind_Resume

Name

`_Unwind_Resume` — private C++ error handling method

Synopsis

```
void _Unwind_Resume(struct _Unwind_Exception * object);
```

Description

`_Unwind_Resume()` resumes propagation of an existing exception *`object`*. A call to this routine is inserted as the end of a landing pad that performs cleanup, but does not resume normal execution. It causes unwinding to proceed further.

_Unwind_Resume_or_Rethrow

Name

_Unwind_Resume_or_Rethrow — private C++ error handling method

Synopsis

```
_Unwind_Reason_Code                    _Unwind_Resume_or_Rethrow(struct
_Unwind_Exception * exception_object);
```

Description

If the unwind was initiated due to a forced unwind, _Unwind_Resume_or_Rethrow() shall resume that operation, else it shall re-raise the exception.

_Unwind_SetGR

Name

_Unwind_SetGR — private C++ error handling method

Synopsis

```
void _Unwind_SetGR(struct _Unwind_Context * context, int index, uint
value);
```

Description

_Unwind_SetGR() sets the *value* of the register *index*ed for the routine identified by the unwind *context*.

_Unwind_SetIP

Name

_Unwind_SetIP — private C++ error handling method

Synopsis

```
#include <unwind.h>
void _Unwind_SetIP(struct _Unwind_Context * context, _Unwind_Ptr
value);
```

Description

_Unwind_SetIP() sets the instruction pointer for the routine identified by the unwind *context* to *value*.

14.15 Interfaces for libdl

Table 14-57 defines the library name and shared object name for the libdl library

Table 14-57 libdl Definition

Library:	libdl
SONAME:	libdl.so.2

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

[LSB] This Specification
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

14.15.1 Dynamic Loader

14.15.1.1 Interfaces for Dynamic Loader

An LSB conforming implementation shall provide the generic functions for Dynamic Loader specified in Table 14-58, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-58 libdl - Dynamic Loader Function Interfaces

dladdr [LSB]	dlclose [SUSv4]	dlerror [SUSv4]	dlopen [LSB]
dlsym [LSB]	dlvsym [LSB]		

14.16 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.

14.16.1 dlfcn.h

```
#define RTLD_NEXT      ((void *) -1l)
#define RTLD_DEFAULT    ((void *) 0)
#define RTLD_LOCAL     0
#define RTLD_LAZY      0x00001
#define RTLD_NOW       0x00002
#define RTLD_NOLOAD    0x00004
#define RTLD_DEEPBIND   0x00008
#define RTLD_GLOBAL    0x00100
#define RTLD_NODELETE   0x01000

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);
extern void *dlvsym(void *handle, const char *name, const char
*version);
```

14.17 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 14.15 shall behave as described in the referenced base document.

dladdr

Name

`dladdr` — find the shared object containing a given address

Synopsis

```
#include <dlfcn.h>

typedef struct {
          const char  *dli_fname;
          void        *dli_fbase;
          const char  *dli_sname;
          void        *dli_saddr;
```



```
} Dl_info;

int dladdr(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

```
#include <dlfcn.h>

void * dlopen(const char * filename, int flag);
```

Description

The `dlopen()` function shall behave as specified in POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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.

Additional flags

In addition to the available values for *`flag`* as documented in POSIX 1003.1-2008 (ISO/IEC 9945-2009), the following values may also be ORed into *`flag`*:

RTLD_NODELETE

Do not unload the library during `dlclose()`. Consequently, the library's static variables are not reinitialized if the library is reloaded with `dlopen()` at a later time.

RTLD_NOLOAD

Do not load the library. This can be used to test if the library is already resident. `dlopen()` returns a `NULL` pointer if it is not resident; it returns the library's handle if it is resident. This flag can also be used to promote the flags on a library that is already loaded. For example, a library that was previously loaded with `RTLD_LOCAL` can be reopened using `RTLD_NOLOAD|RTLD_GLOBAL`.

RTLD_DEEPBIND

Place the lookup scope of the symbols in this library ahead of the global scope. This means that a self-contained library will use its own symbols in preference to global symbols with the same name contained in libraries that have already been loaded.

dlsym

Name

`dlsym` — obtain the address of a symbol from a dlopen object

Description

`dlsym()` is as specified in the POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), are required, with behavior as described in POSIX 1003.1-2008 (ISO/IEC 9945-2009).

dlvsym

Name

`dlvsym` — obtain the address of a symbol from a dlopen object

Synopsis

```
#include <dlfcn.h>
void * dlvsym(void * handle, char * name, char * version);
```

Description

`dlvsym()` does the same as dlsym() but takes a version string as an additional argument.

14.18 Interfaces for librt

Table 14-59 defines the library name and shared object name for the librt library

Table 14-59 librt Definition

Library:	librt
SONAME:	librt.so.1

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

[LFS] Large File Support
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

14.18.1 Shared Memory Objects

14.18.1.1 Interfaces for Shared Memory Objects

An LSB conforming implementation shall provide the generic functions for Shared Memory Objects specified in Table 14-60, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-60 librt - Shared Memory Objects Function Interfaces

shm_open [SUSv4]	shm_unlink [SUSv4]		

14.18.2 Asynchronous I/O

14.18.2.1 Interfaces for Asynchronous I/O

An LSB conforming implementation shall provide the generic functions for Asynchronous I/O specified in Table 14-61, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-61 librt - Asynchronous I/O Function Interfaces

aio_cancel [SUSv4]	aio_cancel64 [LFS]	aio_error [SUSv4]	aio_error64 [LFS]
aio_fsync [SUSv4]	aio_fsync64 [LFS]	aio_read [SUSv4]	aio_read64 [LFS]
aio_return [SUSv4]	aio_return64 [LFS]	aio_suspend [SUSv4]	aio_suspend64 [LFS]
aio_write [SUSv4]	aio_write64 [LFS]	lio_listio(GLIBC_2.4) [SUSv4]	lio_listio64(GLIBC_2.4) [LFS]

14.18.3 Clock

14.18.3.1 Interfaces for Clock

An LSB conforming implementation shall provide the generic functions for Clock specified in Table 14-62, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-62 librt - Clock Function Interfaces

clock_getcpuclockid [SUSv4]	clock_getres [SUSv4]	clock_gettime [SUSv4]	clock_nanosleep [SUSv4]
clock_settime [SUSv4]			

14.18.4 Timers

14.18.4.1 Interfaces for Timers

An LSB conforming implementation shall provide the generic functions for Timers specified in Table 14-63, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-63 librt - Timers Function Interfaces

timer_create [SUSv4]	timer_delete [SUSv4]	timer_getoverrun [SUSv4]	timer_gettime [SUSv4]
timer_settime [SUSv4]			

14.18.5 Message Queues

14.18.5.1 Interfaces for Message Queues

An LSB conforming implementation shall provide the generic functions for Message Queues specified in Table 14-64, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-64 librt - Message Queues Function Interfaces

mq_close(GLIBC_2.3.4) [SUSv4]	mq_getattr(GLIBC_2.3.4) [SUSv4]	mq_notify(GLIBC_2.3.4) [SUSv4]	mq_open(GLIBC_2.3.4) [SUSv4]
mq_receive(GLIBC_2.3.4) [SUSv4]	mq_send(GLIBC_2.3.4) [SUSv4]	mq_setattr(GLIBC_2.3.4) [SUSv4]	mq_timedreceive(GLIBC_2.3.4) [SUSv4]
mq_timedsend(GLIBC_2.3.4) [SUSv4]	mq_unlink(GLIBC_2.3.4) [SUSv4]		

14.19 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.

14.19.1 aio.h

```
#define AIO_CANCELED    0
#define AIO_NOTCANCELED 1
#define AIO_ALLDONE     2

#define LIO_READ       0
#define LIO_WRITE      1
#define LIO_NOP 2

#define LIO_WAIT       0
#define LIO_NOWAIT     1

struct aiocb {
    int aio_fildes;           /* File desriptor */
    int aio_lio_opcode;        /* Operation to be performed */
    int aio_reqprio;          /* Request priority offset */
    void *aio_buf;            /* Location of buffer */
    size_t aio_nbytes;         /* Length of transfer */
    struct sigevent aio_sigevent;       /* Signal number and value
*/
    struct aiocb *__next_prio;  /* internal, do not use */
    int __abs_prio;           /* internal, do not use */
    int __policy;             /* internal, do not use */
    int __error_code;          /* internal, do not use */
    ssize_t __return_value;     /* internal, do not use */
    off_t aio_offset;          /* File offset */
    char __pad[sizeof(off64_t) - sizeof(off_t)];
    char __unused[32];
};
struct aiocb64 {
    int aio_fildes;           /* File desriptor */
    int aio_lio_opcode;        /* Operation to be performed */
    int aio_reqprio;          /* Request priority offset */
    void *aio_buf;            /* Location of buffer */
    size_t aio_nbytes;         /* Length of transfer */
    struct sigevent aio_sigevent;       /* Signal number and value
*/
    struct aiocb *__next_prio;  /* internal, do not use */
    int __abs_prio;           /* internal, do not use */
    int __policy;             /* internal, do not use */
    int __error_code;          /* internal, do not use */
    ssize_t __return_value;     /* internal, do not use */
    off64_t aio_offset;        /* File offset */
    char __unused[32];
};
extern int aio_cancel(int fildes, struct aiocb *aiocbp);
extern int aio_cancel64(int fildes, struct aiocb64 *aiocbp);
extern int aio_error(struct aiocb *aiocbp);
extern int aio_error64(struct aiocb64 *aiocbp);
extern int aio_fsync(int operation, struct aiocb *aiocbp);
extern int aio_fsync64(int operation, struct aiocb64 *aiocbp);
extern int aio_read(struct aiocb *aiocbp);
extern int aio_read64(struct aiocb64 *aiocbp);
extern int aio_return(struct aiocb *aiocbp);
extern int aio_return64(struct aiocb64 *aiocbp);
extern int aio_suspend(struct aiocb *list[], int nent,
                       struct timespec *timeout);
extern int aio_suspend64(struct aiocb64 *list[], int nent,
                         struct timespec *timeout);
extern int aio_write(struct aiocb *aiocbp);
extern int aio_write64(struct aiocb64 *aiocbp);
extern int lio_listio(int mode, struct aiocb *list[], int nent,
                      struct sigevent *sig);
extern int lio_listio64(int mode, struct aiocb64 *list[], int nent,
```



```
                    struct sigevent *sig);
```

14.19.2 mqueue.h

```
typedef int mqd_t;
struct mq_attr {
    long int mq_flags;
    long int mq_maxmsg;
    long int mq_msgsize;
    long int mq_curmsgs;
    long int __pad[4];
};
extern int mq_close(mqd_t __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);
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);
```

14.20 Interfaces for libcrypt

Table 14-65 defines the library name and shared object name for the libcrypt library

Table 14-65 libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

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

[LSB] This Specification
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

14.20.1 Encryption

14.20.1.1 Interfaces for Encryption

An LSB conforming implementation shall provide the generic functions for Encryption specified in Table 14-66, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-66 libcrypt - Encryption Function Interfaces

crypt [SUSv4]	crypt_r [LSB]	encrypt [SUSv4]	encrypt_r [LSB]
setkey [SUSv4]	setkey_r [LSB]		

14.21 Data Definitions for libcrypt

This section defines global identifiers and their values that are associated with interfaces contained in libcrypt. 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.

14.21.1 crypt.h

```
struct crypt_data {
    char keysched[128];
    char sb0[32768];
    char sb1[32768];
    char sb2[32768];
    char sb3[32768];
    char crypt_3_buf[14];
    char current_salt[2];
    long int current_saltbits;
    int direction;
    int initialized;
};
extern char *crypt_r(const char *key, const char *salt,
                     struct crypt_data *data);
extern void encrypt_r(const char *block, int edflag,
                      struct crypt_data *data);
extern void setkey_r(const char *key, struct crypt_data *data);
```

14.22 Interface Definitions for libcrypt

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

Other interfaces listed in Section 14.20 shall behave as described in the referenced base document.

crypt_r

Name

`crypt_r` — Cryptographic string encoding function

Synopsis

```
#include <crypt.h>
char * crypt_r(const char * key, const char * salt, struct crypt_data
* data);
```

Description

The `crypt_r()` function is a re-entrant version of the `crypt()` function. `crypt_r()` shall behave as specified for `crypt()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with an additional parameter, a pointer to a structure which is used to store result data and bookkeeping information.

The caller should set the *`initialized`* field of the `crypt_data` structure to zero before the first call to `crypt_r()`.

Notes

INSERT TEXT HERE

See Also

`crypt()`, `setkey_r()`, `encrypt_r()`.

encrypt_r

Name

`encrypt_r` — Cryptographic encoding function

Synopsis

```
#include <crypt.h>
void encrypt_r(const char * block, int edflag, struct crypt_data *
data);
```

Description

The `encrypt_r()` function is a re-entrant version of the `encrypt()` function. `encrypt_r()` shall behave as specified for `encrypt()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with an additional parameter, a pointer to a structure which is used to store result data and bookkeeping information.

Notes

INSERT TEXT HERE

See Also

`encrypt()`, `crypt_r()`, `setkey_r()`.

setkey_r

Name

`setkey_r` — Set cryptographic encoding key

Synopsis

```
#include <crypt.h>
void setkey_r(const char * key, struct crypt_data * data);
```

Description

The `setkey_r()` function is a re-entrant version of the `setkey()` function. `setkey_r()` shall behave as specified for `setkey()` in POSIX 1003.1-2008 (ISO/IEC 9945-2009), but with an additional parameter, a pointer to a structure which is used to store result data and bookkeeping information.

The caller should set the *`initialized`* field of the `crypt_data` structure to zero before the first call to `setkey_r()`.

Notes

INSERT TEXT HERE

See Also

`setkey()`, `crypt_r()`, `encrypt_r()`.

14.23 Interfaces for libpam

Table 14-67 defines the library name and shared object name for the libpam library

Table 14-67 libpam Definition

Library:	libpam
SONAME:	libpam.so.0

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
[PAM] PAM

14.23.1 Pluggable Authentication API

14.23.1.1 Interfaces for Pluggable Authentication API

An LSB conforming implementation shall provide the generic functions for Pluggable Authentication API specified in Table 14-68, with the full mandatory functionality as described in the referenced underlying specification.

Table 14-68 libpam - Pluggable Authentication API Function Interfaces

pam_acct_mgmt(LIBPAM_1.0) [LSB]	pam_authenticate(LIBPAM_1.0) [LSB]	pam_chauthtok(LIBPAM_1.0) [LSB]	pam_close_session(LIBPAM_1.0) [LSB]
pam_end(LIBPAM_1.0) [LSB]	pam_fail_delay(LIBPAM_1.0) [LSB]	pam_get_data(LIBPAM_1.0) [PAM]	pam_get_item(LIBPAM_1.0) [LSB]
pam_get_user(LIBPAM_1.0) [PAM]	pam_getenv(LIBPAM_1.0) [LSB]	pam_getenvlist(LIBPAM_1.0) [LSB]	pam_open_session(LIBPAM_1.0) [LSB]
pam_putenv(LIBPAM_1.0) [LSB]	pam_set_data(LIBPAM_1.0) [PAM]	pam_set_item(LIBPAM_1.0) [LSB]	pam_setcred(LIBPAM_1.0) [LSB]
pam_start(LIBPAM_1.0) [LSB]	pam_strerror(LIBPAM_1.0) [LSB]		

14.24 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.

14.24.1 security/_pam_types.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;
};

#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
lock busy */
#define PAM_AUTHTOK_DISABLE_AGING       23       /* 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 */

#define PAM_DISALLOW_NULL_AUTHTOK      0x0001U
#define PAM_ESTABLISH_CRED     0x0002U /* Set user credentials for
an authentication service */
#define  PAM_DELETE_CRED  0x0004U  /*  Delete  user  credentials
associated with an authentication se */
#define  PAM_REINITIALIZE_CRED     0x0008U  /*  Reinitialize  user
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_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_putenv(pam_handle_t *, const char *);
extern int pam_set_item(pam_handle_t *, int, const void *);
extern const char *pam_strerror(pam_handle_t *, int);
```

14.24.2 security/pam_appl.h

```
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_open_session(pam_handle_t *, int);
extern int pam_setcred(pam_handle_t *, int);
extern  int  pam_start(const  char  *,  const  char  *,  const  struct
pam_conv *,
                pam_handle_t * *);
```

14.24.3 security/pam_modules.h

```
extern int pam_get_data(const pam_handle_t *, const char *, const
void **);
extern int pam_get_user(pam_handle_t *, const char **, const char
*);
extern int pam_set_data(pam_handle_t *, const char *, void *,
                  void (*)(pam_handle_t *, void *, int));
```

14.25 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 14.23 shall behave as described in the referenced base document.

pam_acct_mgmt

Name

`pam_acct_mgmt` — establish the status of a user's account

Synopsis

```
#include <security/pam_appl.h>
int pam_acct_mgmt(pam_handle_t * pamh, int flags);
```

Description

`pam_acct_mgmt()` establishes the account's usability and the user's accessibility to the system. It is typically called after the user has been authenticated.

`flags` may be specified as any valid flag (namely, one of those applicable to the *`flags`* argument of `pam_authenticate()`). Additionally, the value of *`flags`* may be logically or'd with `PAM_SILENT`.

Return Value

PAM_SUCCESS

Success.

PAM_NEW_AUTHTOK_REQD

User is valid, but user's authentication token has expired. The correct response to this return-value is to require that the user satisfy the `pam_chauthtok()` function before obtaining service. It may not be possible for an application to do this. In such a case, the user should be denied access until the account password is updated.

PAM_ACCT_EXPIRED

User is no longer permitted access to the system.

PAM_AUTH_ERR

Authentication error.

PAM_PERM_DENIED

User is not permitted to gain access at this time.

PAM_USER_UNKNOWN

User is not known to a module's account management component.

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 * pamh, int flags);
```

Description

`pam_authenticate()` serves as an interface to the authentication mechanisms of the loaded modules.

`flags` is an optional parameter that may be specified by the following value:

PAM_DISALLOW_NULL_AUTHTOK

Instruct the authentication modules to return `PAM_AUTH_ERR` if the user does not have a registered authorization token.

Additionally, the value of *`flags`* may be logically or'd with `PAM_SILENT`.

The process may need to be privileged in order to successfully call this function.

Return Value

PAM_SUCCESS

Success.

PAM_AUTH_ERR

User was not authenticated or process did not have sufficient privileges to perform authentication.

PAM_CRED_INSUFFICIENT

Application does not have sufficient credentials to authenticate the user.

PAM_AUTHINFO_UNAVAIL

Modules were not able to access the authentication information. This might be due to a network or hardware failure, etc.

PAM_USER_UNKNOWN

Supplied username is not known to the authentication service.

PAM_MAXTRIES

One or more authentication modules has reached its limit of tries authenticating the user. Do not try again.

PAM_ABORT

One or more authentication modules failed to load.

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

```
#include <security/pam_appl.h>
int pam_end(pam_handle_t * pamh, int pam_status);
```

Description

`pam_end()` terminates use of the PAM library. On success, the contents of `*pamh` are no longer valid, and all memory associated with it is invalid.

Normally, `pam_status` is passed the value `PAM_SUCCESS`, but in the event of an unsuccessful service application, the appropriate PAM error return value should be used.

Return Value

PAM_SUCCESS

Success.

Note: Errors may be translated to text with `pam_strerror()`.

pam_fail_delay

Name

`pam_fail_delay` — specify delay time to use on authentication error

Synopsis

```
#include <security/pam_appl.h>
int pam_fail_delay(pam_handle_t * pamh, unsigned int micro_sec);
```

Description

`pam_fail_delay()` specifies the minimum delay for the PAM library to use when an authentication error occurs. The actual delay can vary by as much at 25%. If this function is called multiple times, the longest time specified by any of the call will be used.

The delay is invoked if an authentication error occurs during the `pam_authenticate()` or `pam_chauthtok()` function calls.

Independent of the success of `pam_authenticate()` or `pam_chauthtok()`, the delay time is reset to its default value of 0 when the PAM library returns control to the application from these two functions.

Return Value

PAM_SUCCESS

Success.

Note: Errors may be translated to text with `pam_strerror()`.

pam_get_item

Name

pam_get_item — obtain the value of the indicated item.

Synopsis

```
#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 environment variable cannot be found, a null pointer shall be returned. The application shall ensure that it does not modify the string pointed to by the pam_getenv() function.

Return Value

On success, pam_getenv() returns a pointer to a string of the form name=value.

pam_getenvlist

Name

`pam_getenvlist` — returns a pointer to the complete PAM environment.

Synopsis

```
#include <security/pam_appl.h>
char * const * pam_getenvlist(pam_handle_t * pamh);
```

Description

`pam_getenvlist()` returns a pointer to the complete PAM environment. This pointer points to an array of pointers to `NUL`-terminated strings and must be terminated by a `NULL` pointer. Each string has the form "name=value".

The PAM library module allocates memory for the returned value and the associated strings. The calling application is responsible for freeing this memory.

Return Value

`pam_getenvlist()` returns an array of string pointers containing the PAM environment. On error, `NULL` is returned.

pam_open_session

Name

`pam_open_session` — indicate session has started

Synopsis

```
#include <security/pam_appl.h>
int pam_open_session(pam_handle_t * pamh, int flags);
```

Description

The `pam_open_session()` function is used to indicate that an authenticated session has begun, after the user has been identified (see `pam_authenticate()`) and, if necessary, granted credentials (see `pam_setcred()`). It is used to inform the module that the user is currently in a session. It should be possible for the PAM library to open a session and close the same session from different applications.

`flags` may have the value `PAM_SILENT` to indicate that no output be generated as a result of this function call.

Return Value

PAM_SUCCESS

Success.

PAM_SESSION_ERR

One of the loaded modules was unable to open a session for the user.

Note: Errors may be translated to text with `pam_strerror()`.

pam_putenv

Name

`pam_putenv` — Add, replace or delete a PAM environment variable

Synopsis

```
#include <security/pam_appl.h>
int pam_putenv(const pam_handle_t * pamh, const char * name_value);
```

Description

The `pam_putenv()` function shall modify the environment list associated with *`pamh`*. If *`name_value`* contains an `'='` character, the characters to the left of the first `'='` character represent the *`name`*, and the remaining characters after the `'='` represent the *`value`*.

If the *`name`* environment variable exists in the environment associated with *`pamh`*, it shall be modified to have the value *`value`*. Otherwise, the *`name`* shall be added to the environment associated with *`pamh`* with the value *`value`*.

If there is no `'='` character in *`name_value`*, the variable in the environment associated with *`pamh`* named *`name_value`* shall be deleted.

Return Value

On success, the `pam_putenv()` function shall return PAM_SUCCESS. Otherwise the return value indicates the error:

PAM_PERM_DENIED

The *`name_value`* argument is a null pointer.

PAM_BAD_ITEM

The PAM environment varable named *`name_value`* does not exist and therefore cannot be deleted.

PAM_ABORT

The PAM handle identifed by *`pamh`* is corrupt.

PAM_BUF_ERR

Memory buffer error.

pam_set_item

Name

`pam_set_item` — (re)set the value of an item.

Synopsis

```
#include <security/pam_appl.h>
int pam_set_item(pam_handle_t * pamh, int item_type, const void * item);
```

Description

`pam_set_item()` (re)sets the value of one of the following item_types:

PAM_SERVICE

service name

PAM_USER

user name

PAM_TTY

terminal name

The value for a device file should include the `/dev/` prefix. The value for graphical, X-based, applications should be the `$DISPLAY` variable.

PAM_RHOST

remote host name

PAM_CONV

conversation structure

PAM_RUSER

remote user name

PAM_USER_PROMPT

string to be used when prompting for a user's name

The default value for this string is `Please enter username: .`

For all *`item_types`* other than `PAM_CONV`, *`item`* is a pointer to a `NULL`-terminated character string. In the case of `PAM_CONV`, *`item`* points to an initialized `pam_conv` structure.

Return Value

PAM_SUCCESS

Success.

PAM_PERM_DENIED

An attempt was made to replace the conversation structure with a `NULL` value.

PAM_BUF_ERR

Function ran out of memory making a copy of the item.

PAM_BAD_ITEM

Application attempted to set an undefined item.

Note: Errors may be translated to text with `pam_strerror()`.

pam_setcred

Name

`pam_setcred` — set the module-specific credentials of the user

Synopsis

```
#include <security/pam_appl.h>
extern int pam_setcred(pam_handle_t * pamh, int flags);
```

Description

`pam_setcred()` sets the module-specific credentials of the user. It is usually called after the user has been authenticated, after the account management function has been called and after a session has been opened for the user.

`flags` 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,

```
struct pam_conv {
           int (*conv) (int num_msg,
                        const struct pam_message * *msg,
                        struct pam_response * *resp,
                        void *appdata_ptr);
           void *appdata_ptr;
```

```
};
```

It is initialized by the application before it is passed to the library. The contents of this structure are attached to the **pamh* handle. The point of this argument is to provide a mechanism for any loaded module to interact directly with the application program; this is why it is called a conversation structure.

When a module calls the referenced `conv()` function, *appdata_ptr* is set to the second element of this structure.

The other arguments of a call to `conv()` concern the information exchanged by module and application. *num_msg* holds the length of the array of pointers passed via *msg*. On success, the pointer *resp* points to an array of *num_msg* `pam_response` structures, holding the application-supplied text. Note that *resp* is a struct `pam_response` array and not an array of pointers.

Return Value

PAM_SUCCESS

Success.

PAM_BUF_ERR

Memory allocation error.

PAM_ABORT

Internal failure.

ERRORS

May be translated to text with `pam_strerror()`.

pam_strerror

Name

`pam_strerror` — returns a string describing the PAM error

Synopsis

```
#include <security/pam_appl.h>
const char * pam_strerror(pam_handle_t * pamh, int errnum);
```

Description

`pam_strerror()` returns a string describing the PAM error associated with *errnum*.

Return Value

On success, this function returns a description of the indicated error. The application should not free or modify this string. 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

15 Utility Libraries

15.1 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
- libncurses
- libncursesw
- libutil

The structure of the definitions for these libraries follows the same model as used for Base Libraries.

15.2 Interfaces for libz

Table 15-1 defines the library name and shared object name for the libz library

Table 15-1 libz Definition

Library:	libz
SONAME:	libz.so.1

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

[LSB] This Specification

15.2.1 Compression Library

15.2.1.1 Interfaces for Compression Library

An LSB conforming implementation shall provide the generic functions for Compression Library specified in Table 15-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 15-2 libz - Compression Library Function Interfaces

adler32 [LSB]	compress [LSB]	compress2 [LSB]	compressBound(ZLIB_1.2.0) [LSB]
crc32 [LSB]	deflate [LSB]	deflateBound(ZL IB_1.2.0) [LSB]	deflateCopy [LSB]
deflateEnd [LSB]	deflateInit2_ [LSB]	deflateInit_ [LSB]	deflateParams [LSB]
deflatePrime(ZLI B_1.2.0.8) [LSB]	deflateReset [LSB]	deflateSetDiction ary [LSB]	get_crc_table [LSB]
gzclearerr(ZLIB_ 1.2.0.2) [LSB]	gzclose [LSB]	gzdopen [LSB]	gzeof [LSB]

gzerror [LSB]	gzflush [LSB]	gzgetc [LSB]	gzgets [LSB]
gzopen [LSB]	gzprintf [LSB]	gzputc [LSB]	gzputs [LSB]
gzread [LSB]	gzrewind [LSB]	gzseek [LSB]	gzsetparams [LSB]
gztell [LSB]	gzwrite [LSB]	inflate [LSB]	inflateBack(ZLIB_1.2.0) [LSB]
inflateBackEnd(ZLIB_1.2.0) [LSB]	inflateBackInit_(ZLIB_1.2.0) [LSB]	inflateCopy(ZLIB_1.2.0) [LSB]	inflateEnd [LSB]
inflateInit2_ [LSB]	inflateInit_ [LSB]	inflateReset [LSB]	inflateSetDictionary [LSB]
inflateSync [LSB]	inflateSyncPoint [LSB]	uncompress [LSB]	zError [LSB]
zlibVersion [LSB]			

15.3 Data Definitions for libz

This section defines global identifiers and their values that are associated with interfaces contained in libz. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

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

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

15.3.1 zconf.h

```
#define ZEXPORT
#define ZEXPORTVA
#define OF(args)       args
#define ZEXTERN extern
```

15.3.2 zlib.h

```
#define ZLIB_VERSION   "1.2.2"
#define Z_NULL  0
#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),(st
rategy),ZLIB_VERSION,sizeof(z_stream))
#define deflateInit(strm,level) \
       deflateInit_((strm),   (level),                ZLIB_VERSION,
sizeof(z_stream))
#define inflateInit2(strm,windowBits)   \
       inflateInit2_((strm),      (windowBits),      ZLIB_VERSION,
sizeof(z_stream))
#define inflateInit(strm)      \
       inflateInit_((strm),                           ZLIB_VERSION,
sizeof(z_stream))
#define inflateBackInit(strm, windowBits, window)      \
       inflateBackInit_((strm), (windowBits), (window), \
                                      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 unsigned int (*in_func) (void *, unsigned char **);
typedef int (*out_func) (void *, unsigned char *, unsigned int);

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 uLong deflateBound(z_streamp strm, uLong sourceLen);
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 deflatePrime(z_streamp strm, int bits, int value);
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 void gzclearerr(gzFile file);
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 int gzgetc(gzFile file);
extern char *gzgets(gzFile file, char *buf, int len);
extern gzFile gzopen(const char *path, const char *mode);
extern int gzprintf(gzFile file, const char *format, ...);
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, voidpc buf, unsigned int len);
extern int inflate(z_streamp strm, int flush);
extern int inflateBack(z_streamp strm, in_func in, void *in_desc,
                       out_func out, void *out_desc);
extern int inflateBackEnd(z_streamp strm);
extern int inflateBackInit_(z_streamp strm, int windowBits,
                            unsigned char *window, const char *version,
                            int stream_size);
extern int inflateCopy(z_streamp dest, 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, const char *version,
                        int stream_size);
extern int inflateReset(z_streamp strm);
extern int inflateSetDictionary(z_streamp strm, 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);
extern const char *zError(int);
extern const char *zlibVersion(void);
```

15.4 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 15.2 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:

```
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`*, and update *`next_out`*, *`avail_out`* and *`total_out`* to reflect the compressed data that has been placed there. If *`flush`* is not `Z_NO_FLUSH`, and *`avail_out`* indicates that there is still space in output buffer, this action shall always occur (see below for further details).

The `deflate()` function shall return when either *`avail_in`* reaches zero (indicating that all the input data has been compressed), or *`avail_out`* reaches zero (indicating that the output buffer is full).

On success, the `deflate()` function shall set the *`adler`* field of the *`stream`* to the `adler32()` checksum of all the input data compressed so far (represented by *`total_in`*).

If the deflate() function shall attempt to determine the type of input data, and set field *data_type* in *stream* to Z_ASCII if the majority of the data bytes fall within the ASCII (ISO 646) printable character range. Otherwise, it shall set *data_type* to Z_BINARY. This data type is informational only, and does not affect the compression algorithm.

> **Note:** Future versions of the LSB may remove this requirement, since it is based on an outdated character set that does not support Internationalization, and does not affect the algorithm. It is included for information only at this release. Applications should not depend on this field.

Flush Operation

The parameter *flush* determines when compressed bits are added to the output buffer in *next_out*. If *flush* is Z_NO_FLUSH, deflate() may return with some data pending output, and not yet added to the output buffer.

If *flush* is Z_SYNC_FLUSH, deflate() shall flush all pending output to *next_out* and align the output to a byte boundary. A synchronization point is generated in the output.

If *flush* is Z_FULL_FLUSH, all output shall be flushed, as for Z_SYNC_FLUSH, and the compression state shall be reset. A synchronization point is generated in the output.

> **Rationale:** Z_SYNC_FLUSH is intended to ensure that the compressed data contains all the data compressed so far, and allows a decompressor to reconstruct all of the input data. Z_FULL_FLUSH allows decompression to restart from this point if the previous compressed data has been lost or damaged. Flushing is likely to degrade the performance of the compression system, and should only be used where necessary.

If *flush* is set to Z_FINISH, all pending input shall be processed and deflate() shall return with Z_STREAM_END if there is sufficient space in the output buffer at *next_out*, as indicated by *avail_out*. If deflate() is called with *flush* set to Z_FINISH and there is insufficient space to store the compressed data, and no other error has occurred during compression, deflate() shall return Z_OK, and the application should call deflate() again with *flush* unchanged, and having updated *next_out* and *avail_out*.

If all the compression is to be done in a single step, deflate() may be called with *flush* set to Z_FINISH immediately after the stream has been initialized if *avail_out* is set to at least the value returned by deflateBound().

Return Value

On success, deflate() shall return Z_OK, unless *flush* was set to Z_FINISH and there was sufficient space in the output buffer to compress all of the input data. In this case, deflate() shall return Z_STREAM_END. On error, deflate() shall return a value to indicate the error.

> **Note:** If deflate() returns Z_OK and has set *avail_out* to zero, the function should be called again with the same value for *flush*, and with updated *next_out* and *avail_out* until deflate() returns with Z_OK (or Z_STREAM_END if *flush* is set to Z_FINISH) and a non-zero *avail_out*.

Errors

On error, `deflate()` shall return a value as described below, and set the *msg* field of *stream* to point to a string describing the error:

`Z_BUF_ERROR`

No progress is possible; either *avail_in* or *avail_out* was zero.

`Z_MEM_ERROR`

Insufficient memory.

`Z_STREAM_ERROR`

The state (as represented in *stream*) is inconsistent, or *stream* was `NULL`.

deflateBound

Name

`deflateBound` — compute compressed data size

Synopsis

```
#include <zlib.h>
int deflateBound(z_streamp stream, uLong sourceLen);
```

Description

The `deflateBound()` function shall estimate the size of buffer required to compress *sourceLen* bytes of data. If successful, the value returned shall be an upper bound for the size of buffer required to compress *sourceLen* bytes of data, using the parameters stored in *stream*, in a single call to `deflate()` with flush set to `Z_FINISH`.

On entry, *stream* should have been initialized via a call to `deflateInit_()` or `deflateInit2_()`.

Return Value

The `deflateBound()` shall return a value representing the upper bound of an array to allocate to hold the compressed data in a single call to `deflate()`. If the *stream* is not correctly initialized, or is `NULL`, then `deflateBound()` may return a conservative value that may be larger than *sourceLen*.

Errors

None defined.

deflateCopy

Name

`deflateCopy` — copy compression stream

Synopsis

```
#include <zlib.h>
int deflateCopy(z_streamp dest, z_streamp source);
```

Description

The `deflateCopy()` function shall copy the compression state information in *`source`* to the uninitialized `z_stream` structure referenced by *`dest`*.

On successful return, *`dest`* will be an exact copy of the stream referenced by *`source`*. The input and output buffer pointers in *`next_in`* and *`next_out`* will reference the same data.

Return Value

On success, `deflateCopy()` shall return Z_OK. Otherwise it shall return a value less than zero to indicate the error.

Errors

On error, `deflateCopy()` shall return a value as described below:

`Z_STREAM_ERROR`

The state in *`source`* is inconsistent, or either *`source`* or *`dest`* was `NULL`.

`Z_MEM_ERROR`

Insufficient memory available.

Application Usage (informative)

This function can be useful when several compression strategies will be tried, for example when there are several ways of pre-processing the input data with a filter. The streams that will be discarded should then be freed by calling `deflateEnd()`. Note that `deflateCopy()` duplicates the internal compression state which can be quite large, so this strategy may be slow and can consume lots of memory.

deflateEnd

Name

`deflateEnd` — free compression stream state

Synopsis

```
#include <zlib.h>
int deflateEnd(z_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_streamp strm, int level, int method, int
windowBits, int memLevel, int strategy, char * version, int stream_size);
```

Description

The deflateInit2_() function shall initialize the compression system. On entry, *strm* shall refer to a user supplied z_stream object (a z_stream_s structure). The following fields shall be set on entry:

zalloc

a pointer to an alloc_func function, used to allocate state information. If this is NULL, a default allocation function will be used.

zfree

a pointer to a free_func function, used to free memory allocated by the *zalloc* function. If this is NULL a default free function will be used.

opaque

If *alloc_func* is not NULL, *opaque* is a user supplied pointer to data that will be passed to the *alloc_func* and *free_func* functions.

If the *version* requested is not compatible with the version implemented, or if the size of the z_stream_s structure provided in *stream_size* does not match the size in the library implementation, deflateInit2_() shall fail, and return Z_VERSION_ERROR.

The *level* supplied shall be a value between 0 and 9, or the value Z_DEFAULT_COMPRESSION. A *level* of 1 requests the highest speed, while a *level* of 9 requests the highest compression. A *level* of 0 indicates that no compression should be used, and the output shall be the same as the input.

The *method* selects the compression algorithm to use. LSB conforming implementation shall support the Z_DEFLATED method, and may support other implementation defined methods.

The *windowBits* parameter shall be a base 2 logarithm of the window size to use, and shall be a value between 8 and 15. A smaller value will use less memory, but will result in a poorer compression ratio, while a higher value will give better compression but utilize more memory.

The *memLevel* parameter specifies how much memory to use for the internal state. The value of *memLevel* shall be between 1 and MAX_MEM_LEVEL. Smaller values use less memory but are slower, while higher values use more memory to gain compression speed.

The *strategy* parameter selects the compression strategy to use:

Z_DEFAULT_STRATEGY

use the system default compression strategy. `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.

The `deflateInit2_()` function is not in the source standard; it is only in the binary standard. Source applications should use the `deflateInit2()` macro.

Return Value

On success, the `deflateInit2_()` function shall return `Z_OK`. Otherwise, `deflateInit2_()` shall return a value as described below to indicate the error.

Errors

On error, `deflateInit2_()` 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.

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_streamp stream, int level, int strategy);
```

Description

The `deflateParams()` function shall dynamically alter the compression parameters for the compression stream object *`stream`*. On entry, *`stream`* shall refer to a user supplied z_stream object (a `z_stream_s` structure), already initialized via a call to `deflateInit_()` or `deflateInit2_()`.

The *`level`* supplied shall be a value between `0` and `9`, or the value `Z_DEFAULT_COMPRESSION`. A *`level`* of `1` requests the highest speed, while a *`level`* of `9` requests the highest compression. A *`level`* of `0` indicates that no compression should be used, and the output shall be the same as the input. If the compression level is altered by `deflateParams()`, and some data has already been compressed with this *`stream`* (i.e. *`total_in`* is not zero), and the new *`level`* requires a different underlying compression method, then *`stream`* shall be flushed by a call to `deflate()`.

The *`strategy`* parameter selects the compression strategy to use:

`Z_DEFAULT_STRATEGY`

: use the system default compression strategy. `Z_DEFAULT_STRATEGY` is particularly appropriate for text data.

`Z_FILTERED`

: use a compression strategy tuned for data consisting largely of small values with a fairly random distribution. `Z_FILTERED` uses more Huffman encoding and less string matching than `Z_DEFAULT_STRATEGY`.

`Z_HUFFMAN_ONLY`

: force Huffman encoding only, with no string match.

Return Value

On success, the `deflateParams()` function shall return `Z_OK`. Otherwise, `deflateParams()` shall return a value as described below to indicate the error.

Errors

On error, `deflateParams()` shall return one of the following error indicators:

`Z_STREAM_ERROR`

: Invalid parameter.

`Z_MEM_ERROR`

: Insufficient memory available.

`Z_BUF_ERROR`

Insufficient space in *`stream`* to flush the current output.

In addition, the *`msg`* field of the *`strm`* may be set to an error message.

Application Usage (Informative)

Applications should ensure that the *`stream`* is flushed, e.g. by a call to **`deflate(stream, Z_SYNC_FLUSH)`** before calling `deflateParams()`, or ensure that there is sufficient space in *`next_out`* (as identified by *`avail_out`*) to ensure that all pending output and all uncompressed input can be flushed in a single call to `deflate()`.

> **Rationale:** Although the `deflateParams()` function should flush pending output and compress all pending input, the result is unspecified if there is insufficient space in the output buffer. Applications should only call `deflateParams()` when the *`stream`* is effectively empty (flushed).
>
> The `deflateParams()` can be used to switch between compression and straight copy of the input data, or to switch to a different kind of input data requiring a different strategy.

deflateReset

Name

`deflateReset` — reset compression stream state

Synopsis

```
#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_streamp stream, const Bytef * dictionary,
uInt dictlen);
```

Description

The `deflateSetDictionary()` function shall initialize the compression dictionary associated with *`stream`* using the *`dictlen`* bytes referenced by *`dictionary`*.

The implementation may silently use a subset of the provided dictionary if the dictionary cannot fit in the current window associated with *`stream`* (see `deflateInit2_()`). The application should ensure that the dictionary is sorted such that the most commonly used strings occur at the end of the dictionary.

If the dictionary is successfully set, the Adler32 checksum of the entire provided dictionary shall be stored in the *`adler`* member of *`stream`*. This value may be used by the decompression system to select the correct dictionary. The compression and decompression systems must use the same dictionary.

`stream` shall reference an initialized compression stream, with *`total_in`* zero (i.e. no data has been compressed since the stream was initialized).

Return Value

On success, `deflateSetDictionary()` shall return Z_OK. Otherwise it shall return Z_STREAM_ERROR to indicate an error.

Errors

On error, `deflateSetDictionary()` shall return a value as described below:

`Z_STREAM_ERROR`

The state in *`stream`* is inconsistent, or *`stream`* was `NULL`.

Application Usage (informative)

The application should provide a dictionary consisting of strings {{{ed note: do we really mean "strings"? Null terminated?}}} that are likely to be encountered in the data to be compressed. The application should ensure that the dictionary is sorted such that the most commonly used strings occur at the end of the dictionary.

The use of a dictionary is optional; however if the data to be compressed is relatively short and has a predictable structure, the use of a dictionary can substantially improve the compression ratio.

get_crc_table

Name

`get_crc_table` — generate a table for crc calculations

Synopsis

```
#include <zlib.h>
const uLongf * get_crc_table(void);
```

Description

Generate tables for a byte-wise 32-bit CRC calculation based on the polynomial: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$

In a multi-threaded application, `get_crc_table()` should be called by one thread to initialize the tables before any other thread calls any `libz` function.

Return Value

The `get_crc_table()` function shall return a pointer to the first of a set of tables used internally to calculate CRC-32 values (see `crc32()`).

Errors

None defined.

gzclose

Name

`gzclose` — close a compressed file stream

Synopsis

```
#include <zlib.h>
 int gzclose (gzFile file );
```

Description

The `gzclose()` function shall close the compressed file stream *`file`*. If *`file`* was open for writing, `gzclose()` shall first flush any pending output. Any state information allocated shall be freed.

Return Value

On success, `gzclose()` shall return Z_OK. Otherwise, `gzclose()` shall return an error value as described below.

Errors

On error, `gzclose()` may set the global variable `errno` to indicate the error. The `gzclose()` shall return a value other than `Z_OK` on error.

`Z_STREAM_ERROR`

`file` was `NULL` (or `Z_NULL`), or did not refer to an open compressed file stream.

`Z_ERRNO`

An error occurred in the underlying base libraries, and the application should check `errno` for further information.

`Z_BUF_ERROR`

no compression progress is possible during buffer flush (see `deflate()`).

gzdopen

Name

`gzdopen` — open a compressed file

Synopsis

```
#include <zlib.h>
 gzFile gzdopen ( int fd, const char *mode );
```

Description

The `gzdopen()` function shall attempt to associate the open file referenced by *`fd`* with a gzFile object. The *`mode`* argument is based on that of `fopen()`, but the *`mode`* parameter may also contain the following characters:

`digit`

> set the compression level to *`digit`*. A low value (e.g. 1) means high speed, while a high value (e.g. 9) means high compression. A compression level of 0 (zero) means no compression. See `deflateInit2_()` for further details.

`[fhR]`

> set the compression strategy to *`[fhR]`*. The letter `f` corresponds to filtered data, the letter `h` corresponds to Huffman only compression, and the letter `R` corresponds to Run Length Encoding. See `deflateInit2_()` for further details.

If *`fd`* refers to an uncompressed file, and *`mode`* refers to a read mode, `gzdopen()` shall attempt to open the file and return a gzFile object suitable for reading directly from the file without any decompression.

If *`mode`* is `NULL`, or if *`mode`* does not contain one of `r`, `w`, or `a`, `gzdopen()` shall return `Z_NULL`, and need not set any other error condition.

Example

```
gzdopen(fileno(stdin), "r");
```

Attempt to associate the standard input with a gzFile object.

Return Value

On success, `gzdopen()` shall return a gzFile object. On failure, `gzdopen()` shall return `Z_NULL` and may set `errno` accordingly.

> **Note:** At version 1.2.2, `zlib` does not set `errno` for several error conditions. Applications may not be able to determine the cause of an error.

Errors

On error, `gzdopen()` may set the global variable `errno` to indicate the error.

gzeof

Name

`gzeof` — check for end-of-file on a compressed file stream

Synopsis

```
#include <zlib.h>
 int gzeof (gzFile file );
```

Description

The `gzeof()` function shall test the compressed file stream *`file`* for end of file.

Return Value

If *`file`* was open for reading and end of file has been reached, `gzeof()` shall return 1. Otherwise, `gzeof()` shall return 0.

Errors

None defined.

gzerror

Name

`gzerror` — decode an error on a compressed file stream

Synopsis

```
#include <zlib.h>
 const char * gzerror (gzFile file, int * errnum);
```

Description

The `gzerror()` function shall return a string describing the last error to have occurred associated with the open compressed file stream referred to by *`file`*. It shall also set the location referenced by *`errnum`* to an integer value that further identifies the error.

Return Value

The `gzerror()` function shall return a string that describes the last error associated with the given *`file`* compressed file stream. This string shall have the format `"%s: %s"`, with the name of the file, followed by a colon, a space, and the description of the error. If the compressed file stream was opened by a call to `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 `gzerror()` to access this error value.

`Z_ERRNO`

> An underlying base library function has indicated an error. The global variable `errno` may be examined for further information.

`Z_STREAM_ERROR`

The stream is invalid, is not open for writing, or is in an invalid state.

`Z_BUF_ERROR`

no compression progress is possible (see `deflate()`).

`Z_MEM_ERROR`

Insufficient memory available to compress.

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 `gzdopen()`).

Return Value

On success, `gzgets()` shall return a pointer to *`buf`*. Otherwise, `gzgets()` shall return Z_NULL. Applications may examine the cause using `gzerror()`.

Errors

On error, `gzgets()` shall return Z_NULL. The following conditions shall always be treated as an error:

`file` is `NULL`, or does not refer to a file open for reading;
`buf` is `NULL`;
`len` is less than or equal to zero.

gzopen

Name

`gzopen` — open a compressed file

Synopsis

```
#include <zlib.h>
 gzFile gzopen (const char *path , const char *mode );
```

Description

The `gzopen()` function shall open the compressed file named by *`path`*. The *`mode`* argument is based on that of `fopen()`, but the *`mode`* parameter may also contain the following characters:

`digit`

> set the compression level to *`digit`*. A low value (e.g. 1) means high speed, while a high value (e.g. 9) means high compression. A compression level of 0 (zero) means no compression. See `deflateInit2_()` for further details.

`[fhR]`

> set the compression strategy to *`[fhR]`*. The letter `f` corresponds to filtered data, the letter `h` corresponds to Huffman only compression, and the letter `R` corresponds to Run Length Encoding. See `deflateInit2_()` for further details.

If *`path`* refers to an uncompressed file, and *`mode`* refers to a read mode, `gzopen()` shall attempt to open the file and return a gzFile object suitable for reading directly from the file without any decompression.

If *`path`* or *`mode`* is `NULL`, or if *`mode`* does not contain one of `r`, `w`, or `a`, `gzopen()` shall return `Z_NULL`, and need not set any other error condition.

The gzFile object is also referred to as a compressed file stream.

Example

```
gzopen("file.gz", "w6h");
```

Attempt to create a new compressed file, `file.gz`, at compression level 6 using Huffman only compression.

Return Value

On success, `gzopen()` shall return a gzFile object (also known as a *compressed file stream*). On failure, `gzopen()` shall return `Z_NULL` and may set `errno` accordingly.

> **Note:** At version 1.2.2, `zlib` does not set `errno` for several error conditions. Applications may not be able to determine the cause of an error.

Errors

On error, `gzopen()` may set the global variable `errno` to indicate the error.

gzprintf

Name

`gzprintf` — format data and compress

Synopsis

```
#include <zlib.h>
 int gzprintf (gzFile file, const char * fmt, ...);
```

Description

The `gzprintf()` function shall format data as for `fprintf()`, and write the resulting string to the compressed file stream *`file`*.

Return Value

The `gzprintf()` function shall return the number of uncompressed bytes actually written, or a value less than or equal to `0` in the event of an error.

Errors

If *`file`* is `NULL`, or refers to a compressed file stream that has not been opened for writing, `gzprintf()` shall return `Z_STREAM_ERROR`. Otherwise, errors are as for `gzwrite()`.

gzputc

Name

`gzputc` — write character to a compressed file

Synopsis

```
#include <zlib.h>
 int gzputc (gzFile file, int c);
```

Description

The `gzputc()` function shall write the single character *`c`*, converted from integer to unsigned character, to the compressed file referenced by *`file`*, which shall have been opened in a write mode (see `gzopen()` and `gzdopen()`).

Return Value

On success, `gzputc()` shall return the value written, otherwise `gzputc()` shall return -1.

Errors

On error, `gzputc()` shall return -1.

gzputs

Name

gzputs — string write to a compressed file

Synopsis

```
#include <zlib.h>
 int gzputs (gzFile file, const char * s);
```

Description

The gzputs() function shall write the null terminated string *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 `gzdopen()`). The `gzread()` function shall read data from *`file`*, and uncompress it into *`buf`*. At most, *`len`* bytes of uncompressed data shall be copied to *`buf`*. If the file is not compressed, `gzread()` shall simply copy data from *`file`* to *`buf`* without alteration.

Return Value

On success, `gzread()` shall return the number of bytes decompressed into *`buf`*. If `gzread()` returns `0`, either the end-of-file has been reached or an underlying read error has occurred. Applications should use `gzerror()` or `gzeof()` to determine which occurred. On other errors, `gzread()` shall return a value less than `0` and applications may examine the cause using `gzerror()`.

Errors

On error, `gzread()` shall set the error number associated with the stream identified by *`file`* to indicate the error. Applications should use `gzerror()` to access this error value.

`Z_ERRNO`

An underlying base library function has indicated an error. The global variable `errno` may be examined for further information.

`Z_STREAM_END`

End of file has been reached on input.

`Z_DATA_ERROR`

A CRC error occurred when reading data; the file is corrupt.

`Z_STREAM_ERROR`

The stream is invalid, or is in an invalid state.

`Z_NEED_DICT`

A dictionary is needed (see `inflateSetDictionary()`).

`Z_MEM_ERROR`

Insufficient memory available to decompress.

gzrewind

Name

`gzrewind` — reset the file-position indicator on a compressed file stream

Synopsis

```
#include <zlib.h>
int gzrewind(gzFile file);
```

Description

The `gzrewind()` function shall set the starting position for the next read on compressed file stream *`file`* to the beginning of file. *`file`* must be open for reading.

`gzrewind()` is equivalent to

```
(int)gzseek(file, 0L, SEEK_SET)
```

.

Return Value

On success, `gzrewind()` shall return 0. On error, `gzrewind()` shall return -1, and may set the error value for *`file`* accordingly.

Errors

On error, `gzrewind()` shall return `-1`, indicating that *`file`* is `NULL`, or does not represent an open compressed file stream, or represents a compressed file stream that is open for writing and is not currently at the beginning of file.

gzseek

Name

`gzseek` — reposition a file-position indicator in a compressed file stream

Synopsis

```
#include <zlib.h>
z_off_t gzseek(gzFile file, z_off_t offset, int whence);
```

Description

The `gzseek()` function shall set the file-position indicator for the compressed file stream *`file`*. The file-position indicator controls where the next read or write operation on the compressed file stream shall take place. The *`offset`* indicates a byte offset in the uncompressed data. The *`whence`* parameter may be one of:

`SEEK_SET`

the offset is relative to the start of the uncompressed data.

`SEEK_CUR`

the offset is relative to the current positition in the uncompressed data.

Note: The value `SEEK_END` need not be supported.

If the *`file`* is open for writing, the new offset must be greater than or equal to the current offset. In this case, `gzseek()` shall compress a sequence of null bytes to fill the gap from the previous offset to the new offset.

Return Value

On success, `gzseek()` shall return the resulting offset in the file expressed as a byte position in the *uncompressed* data stream. On error, `gzseek()` shall return -1, and may set the error value for *`file`* accordingly.

Errors

On error, `gzseek()` shall return -1. The following conditions shall always result in an error:

- *`file`* is `NULL`
- *`file`* does not represent an open compressed file stream.
- *`file`* refers to a compressed file stream that is open for writing, and the newly computed offset is less than the current offset.
- The newly computed offset is less than zero.
- *`whence`* is not one of the supported values.

Application Usage (informative)

If *`file`* is open for reading, the implementation may still need to uncompress all of the data up to the new offset. As a result, `gzseek()` may be extremely slow in some circumstances.

gzsetparams

Name

`gzsetparams` — dynamically set compression parameters

Synopsis

```
#include <zlib.h>
 int gzsetparams (gzFile file, int level, int strategy);
```

Description

The `gzsetparams()` function shall set the compression level and compression strategy on the compressed file stream referenced by *`file`*. The compressed file stream shall have been opened in a write mode. The *`level`* and *`strategy`* are as defined in deflateInit2.. If there is any data pending writing, it shall be flushed before the parameters are updated.

Return Value

On success, the `gzsetparams()` function shall return `Z_OK`.

Errors

On error, `gzsetparams()` shall return one of the following error indications:

`Z_STREAM_ERROR`

Invalid parameter, or *`file`* not open for writing.

`Z_BUF_ERROR`

An internal inconsistency was detected while flushing the previous buffer.

gztell

Name

`gztell` — find position on a compressed file stream

Synopsis

```
#include <zlib.h>
 z_off_t gztell (gzFile file );
```

Description

The `gztell()` function shall return the starting position for the next read or write operation on compressed file stream *`file`*. This position represents the number of bytes from the beginning of file in the uncompressed data.

`gztell()` is equivalent to

```
gzseek(file, 0L, SEEK_CUR)
```

.

Return Value

`gztell()` shall return the current offset in the file expressed as a byte position in the *uncompressed* data stream. On error, `gztell()` shall return -1, and may set the error value for *`file`* accordingly.

Errors

On error, `gztell()` shall return -1, indicating that *`file`* is `NULL`, or does not represent an open compressed file stream.

gzwrite

Name

`gzwrite` — write to a compressed file

Synopsis

```
#include <zlib.h>
 int gzwrite (gzFile file, voidpc buf, unsigned int len);
```

Description

The `gzwrite()` function shall write data to the compressed file referenced by *`file`*, which shall have been opened in a write mode (see `gzopen()` and `gzdopen()`). On entry, *`buf`* shall point to a buffer containing *`len`* bytes of uncompressed data. The `gzwrite()` function shall compress this data and write it to *`file`*. The `gzwrite()` function shall return the number of uncompressed bytes actually written.

Return Value

On success, `gzwrite()` shall return the number of uncompressed bytes actually written to *`file`*. On error `gzwrite()` shall return a value less than or equal to `0`. Applications may examine the cause using `gzerror()`.

Errors

On error, `gzwrite()` shall set the error number associated with the stream identified by *`file`* to indicate the error. Applications should use `gzerror()` to access this error value.

`Z_ERRNO`

An underlying base library function has indicated an error. The global variable `errno` may be examined for further information.

`Z_STREAM_ERROR`

The stream is invalid, is not open for writing, or is in an invalid state.

`Z_BUF_ERROR`

no compression progress is possible (see `deflate()`).

`Z_MEM_ERROR`

Insufficient memory available to compress.

inflate

Name

`inflate` — decompress data

Synopsis

```
#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_streamp stream);
```

Description

The `inflateEnd()` function shall free all allocated state information referenced by *`stream`*. All pending output is discarded, and unprocessed input is ignored.

Return Value

On success, `inflateEnd()` shall return Z_OK. Otherwise it shall return Z_STREAM_ERROR to indicate the error.

Errors

On error, `inflateEnd()` shall return Z_STREAM_ERROR. The following conditions shall be treated as an error:

- The state in *`stream`* is inconsistent.
- *`stream`* is `NULL`.
- The *`zfree`* function pointer is `NULL`.

inflateInit2_

Name

`inflateInit2_` — initialize decompression system

Synopsis

```
#include <zlib.h>
 int inflateInit2_ (z_streamp strm, int windowBits, char * version, int
stream_size);
```

Description

The `inflateInit2_()` function shall initialize the decompression system. On entry, *`strm`* shall refer to a user supplied z_stream object (a `z_stream_s` structure). The following fields shall be set on entry:

`zalloc`

a pointer to an alloc_func function, used to allocate state information. If this is `NULL`, a default allocation function will be used.

`zfree`

a pointer to a free_func function, used to free memory allocated by the *`zalloc`* function. If this is `NULL` a default free function will be used.

`opaque`

If *`alloc_func`* is not `NULL`, *`opaque`* is a user supplied pointer to data that will be passed to the *`alloc_func`* and *`free_func`* functions.

If the *`version`* requested is not compatible with the version implemented, or if the size of the `z_stream_s` structure provided in *`stream_size`* does not match the size in the library implementation, `inflateInit2_()` shall fail, and return `Z_VERSION_ERROR`.

The *`windowBits`* parameter shall be a base 2 logarithm of the maximum window size to use, and shall be a value between `8` and `15`. If the input data was compressed with a larger window size, subsequent attempts to decompress this data will fail with `Z_DATA_ERROR`, rather than try to allocate a larger window.

The `inflateInit2_()` function is not in the source standard; it is only in the binary standard. Source applications should use the `inflateInit2()` macro.

Return Value

On success, the `inflateInit2_()` function shall return `Z_OK`. Otherwise, `inflateInit2_()` shall return a value as described below to indicate the error.

Errors

On error, `inflateInit2_()` shall return one of the following error indicators:

`Z_STREAM_ERROR`

Invalid parameter.

`Z_MEM_ERROR`

Insufficient memory available.

`Z_VERSION_ERROR`

The version requested is not compatible with the library version, or the z_stream size differs from that used by the library.

In addition, the *`msg`* field of the *`strm`* may be set to an error message.

inflateInit_

Name

`inflateInit_` — initialize decompression system

Synopsis

```
#include <zlib.h>
int inflateInit_(z_streamp stream, const char * version, int
stream_size);
```

Description

The `inflateInit_()` function shall initialize the decompression system. On entry, *`stream`* shall refer to a user supplied z_stream object (a `z_stream_s` structure). The following fields shall be set on entry:

`zalloc`

a pointer to an alloc_func function, used to allocate state information. If this is `NULL`, a default allocation function will be used.

`zfree`

a pointer to a free_func function, used to free memory allocated by the *`zalloc`* function. If this is `NULL` a default free function will be used.

`opaque`

If *`alloc_func`* is not `NULL`, *`opaque`* is a user supplied pointer to data that will be passed to the *`alloc_func`* and *`free_func`* functions.

If the *`version`* requested is not compatible with the version implemented, or if the size of the `z_stream_s` structure provided in *`stream_size`* does not match the size in the library implementation, `inflateInit_()` shall fail, and return `Z_VERSION_ERROR`.

The `inflateInit_()` function is not in the source standard; it is only in the binary standard. Source applications should use the `inflateInit()` macro.

The `inflateInit_()` shall be equivalent to

```
inflateInit2_(strm, MAX_WBITS, version, stream_size);
```

Return Value

On success, the `inflateInit_()` function shall return `Z_OK`. Otherwise, `inflateInit_()` shall return a value as described below to indicate the error.

Errors

On error, `inflateInit_()` shall return one of the following error indicators:

`Z_STREAM_ERROR`

Invalid parameter.

`Z_MEM_ERROR`

Insufficient memory available.

`Z_VERSION_ERROR`

The version requested is not compatible with the library version, or the z_stream size differs from that used by the library.

In addition, the *`msg`* field of the *`strm`* may be set to an error message.

inflateReset

Name

`inflateReset` — reset decompression stream state

Synopsis

```
#include <zlib.h>
int inflateReset(z_streamp stream);
```

Description

The `inflateReset()` function shall reset all state associated with *`stream`*. All pending output shall be discarded, and the counts of processed bytes (*`total_in`* and *`total_out`*) shall be reset to zero.

Return Value

On success, `inflateReset()` shall return Z_OK. Otherwise it shall return Z_STREAM_ERROR to indicate the error.

Errors

On error, `inflateReset()` shall return Z_STREAM_ERROR. The following conditions shall be treated as an error:

- The state in *`stream`* is inconsistent or inappropriate.
- *`stream`* is `NULL`.

inflateSetDictionary

Name

`inflateSetDictionary` — initialize decompression dictionary

Synopsis

```
#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_streamp stream);
```

Description

The `inflateSyncPoint()` function shall return a non-zero value if the compressed data stream referenced by *`stream`* is at a synchronization point.

Return Value

If the compressed data in *`stream`* is at a synchronization point (see `deflate()` with a flush level of `Z_SYNC_FLUSH` or `Z_FULL_FLUSH`), `inflateSyncPoint()` shall return a non-zero value, other than `Z_STREAM_ERROR`. Otherwise, if the *`stream`* is valid, `inflateSyncPoint()` shall return 0. If *`stream`* is invalid, or in an invalid state, `inflateSyncPoint()` shall return Z_STREAM_ERROR to indicate the error.

Errors

On error, `inflateSyncPoint()` shall return a value as described below:

`Z_STREAM_ERROR`

The state (as represented in *`stream`*) is inconsistent, or *`stream`* was `NULL`.

uncompress

Name

`uncompress` — uncompress data

Synopsis

```
#include <zlib.h>
int uncompress(Bytef * dest, uLongf * destLen, const Bytef * source,
uLong sourceLen);
```

Description

The `uncompress()` function shall attempt to uncompress *`sourceLen`* bytes of data in the buffer *`source`*, placing the result in the buffer *`dest`*.

On entry, *`destLen`* should point to a value describing the size of the *`dest`* buffer. The application should ensure that this value is large enough to hold the entire uncompressed data.

> **Note:** The LSB does not describe any mechanism by which a compressor can communicate the size required to the uncompressor.

On successful exit, the variable referenced by *`destLen`* shall be updated to hold the length of uncompressed data in *`dest`*.

Return Value

On success, `uncompress()` shall return Z_OK. Otherwise, `uncompress()` shall return a value to indicate the error.

Errors

On error, `uncompress()` shall return a value as described below:

`Z_BUF_ERROR`

The buffer *`dest`* was not large enough to hold the uncompressed data.

`Z_MEM_ERROR`

Insufficient memory.

`Z_DATA_ERROR`

The compressed data (referenced by *`source`*) was corrupted.

zError

Name

`zError` — translate error number to string

Synopsis

```
#include <zlib.h>
const char * zError(int err);
```

Description

The `zError()` function shall return the string identifying the error associated with *err*. This allows for conversion from error code to string for functions such as `compress()` and `uncompress()`, that do not always set the string version of an error.

Return Value

The `zError()` function shall return a the string identifying the error associated with *err*, or NULL if *err* is not a valid error code.

It is unspecified if the string returned is determined by the setting of the `LC_MESSAGES` category in the current locale.

Errors

None defined.

zlibVersion

Name

`zlibVersion` — discover library version at run time

Synopsis

```
#include <zlib.h>
 const char * zlibVersion (void);
```

Description

The `zlibVersion()` function shall return the string identifying the interface version at the time the library was built.

Applications should compare the value returned from `zlibVersion()` with the macro constant `ZLIB_VERSION` for compatibility.

Return Value

The `zlibVersion()` function shall return a the string identifying the version of the library currently implemented.

Errors

None defined.

15.5 Interfaces for libncurses

Table 15-3 defines the library name and shared object name for the libncurses library

Table 15-3 libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

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, Issue 7.

```
extern const char *keyname (int);
extern SCREEN *newterm (const char *, FILE *, FILE *);
extern const char *unctrl (chtype);

extern int mvprintw (int, int, const char *, ...);
extern int mvwprintw (WINDOW *, int, int, const char *, ...);
extern int printw (const char *, ...);
extern int vwprintw (WINDOW *, const char *, va_list);
extern int vw_printw (WINDOW *, const char *, va_list);
extern int wprintw (WINDOW *, const char *, ...);

extern int mvscanw (int, int, const char *, ...);
extern int mvwscanw (WINDOW *, int, int, const char *, ...);
extern int scanw (const char *, ...);
extern int vwscanw (WINDOW *, const char *, va_list);
extern int vw_scanw (WINDOW *, const char *, va_list);
extern int wscanw (WINDOW *, const char *, ...);
```

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

[LSB] This Specification
[X-CURSES] X/Open Curses, Issue 7

15.5.1 Curses

15.5.1.1 Interfaces for Curses

An LSB conforming implementation shall provide the generic functions for Curses specified in Table 15-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 15-4 libncurses - Curses Function Interfaces

addch [X-CURSES]	addchnstr [X-CURSES]	addchstr [X-CURSES]	addnstr [X-CURSES]
addstr [X-CURSES]	attr_get [X-CURSES]	attr_off [X-CURSES]	attr_on [X-CURSES]
attr_set [X-CURSES]	attroff [X-CURSES]	attron [X-CURSES]	attrset [X-CURSES]

baudrate [X-CURSES]	beep [X-CURSES]	bkgd [X-CURSES]	bkgdset [X-CURSES]
border [X-CURSES]	box [X-CURSES]	can_change_color [X-CURSES]	cbreak [X-CURSES]
chgat [X-CURSES]	clear [X-CURSES]	clearok [X-CURSES]	clrtobot [X-CURSES]
clrtoeol [X-CURSES]	color_content [X-CURSES]	color_set [X-CURSES]	copywin [X-CURSES]
curs_set [X-CURSES]	def_prog_mode [X-CURSES]	def_shell_mode [X-CURSES]	del_curterm [X-CURSES]
delay_output [X-CURSES]	delch [X-CURSES]	deleteln [X-CURSES]	delscreen [X-CURSES]
delwin [X-CURSES]	derwin [X-CURSES]	doupdate [X-CURSES]	dupwin [X-CURSES]
echo [X-CURSES]	echochar [X-CURSES]	endwin [X-CURSES]	erase [X-CURSES]
erasechar [X-CURSES]	filter [X-CURSES]	flash [X-CURSES]	flushinp [X-CURSES]
getbkgd [X-CURSES]	getch [X-CURSES]	getnstr [X-CURSES]	getstr [X-CURSES]
getwin [X-CURSES]	halfdelay [X-CURSES]	has_colors [X-CURSES]	has_ic [X-CURSES]
has_il [X-CURSES]	hline [X-CURSES]	idcok [X-CURSES]	idlok [X-CURSES]
immedok [X-CURSES]	inch [X-CURSES]	inchnstr [LSB]	inchstr [LSB]
init_color [X-CURSES]	init_pair [X-CURSES]	initscr [X-CURSES]	innstr [X-CURSES]
insch [X-CURSES]	insdelln [X-CURSES]	insertln [X-CURSES]	insnstr [X-CURSES]
insstr [X-CURSES]	instr [LSB]	intrflush [X-CURSES]	is_linetouched [X-CURSES]
is_wintouched [X-CURSES]	isendwin [X-CURSES]	keyname [X-CURSES]	keypad [X-CURSES]
killchar [X-CURSES]	leaveok [X-CURSES]	longname [X-CURSES]	meta [X-CURSES]
move [X-CURSES]	mvaddch [X-CURSES]	mvaddchnstr [X-CURSES]	mvaddchstr [X-CURSES]

mvaddnstr [X-CURSES]	mvaddstr [X-CURSES]	mvchgat [X-CURSES]	mvcur [LSB]
mvdelch [X-CURSES]	mvderwin [X-CURSES]	mvgetch [X-CURSES]	mvgetnstr [X-CURSES]
mvgetstr [X-CURSES]	mvhline [X-CURSES]	mvinch [X-CURSES]	mvinchnstr [LSB]
mvinchstr [LSB]	mvinnstr [X-CURSES]	mvinsch [X-CURSES]	mvinsnstr [X-CURSES]
mvinsstr [X-CURSES]	mvinstr [LSB]	mvprintw [X-CURSES]	mvscanw [LSB]
mvvline [X-CURSES]	mvwaddch [X-CURSES]	mvwaddchnstr [X-CURSES]	mvwaddchstr [X-CURSES]
mvwaddnstr [X-CURSES]	mvwaddstr [X-CURSES]	mvwchgat [X-CURSES]	mvwdelch [X-CURSES]
mvwgetch [X-CURSES]	mvwgetnstr [X-CURSES]	mvwgetstr [X-CURSES]	mvwhline [X-CURSES]
mvwin [X-CURSES]	mvwinch [X-CURSES]	mvwinchnstr [LSB]	mvwinchstr [LSB]
mvwinnstr [X-CURSES]	mvwinsch [X-CURSES]	mvwinsnstr [X-CURSES]	mvwinsstr [X-CURSES]
mvwinstr [LSB]	mvwprintw [X-CURSES]	mvwscanw [LSB]	mvwvline [X-CURSES]
napms [X-CURSES]	newpad [X-CURSES]	newterm [X-CURSES]	newwin [X-CURSES]
nl [X-CURSES]	nocbreak [X-CURSES]	nodelay [X-CURSES]	noecho [X-CURSES]
nonl [X-CURSES]	noqiflush [X-CURSES]	noraw [X-CURSES]	notimeout [X-CURSES]
overlay [X-CURSES]	overwrite [X-CURSES]	pair_content [X-CURSES]	pechochar [X-CURSES]
pnoutrefresh [X-CURSES]	prefresh [X-CURSES]	printw [X-CURSES]	putp [X-CURSES]
putwin [X-CURSES]	qiflush [X-CURSES]	raw [X-CURSES]	redrawwin [X-CURSES]
refresh [X-CURSES]	reset_prog_mode [X-CURSES]	reset_shell_mode [X-CURSES]	resetty [X-CURSES]
restartterm [X-CURSES]	ripoffline [LSB]	savetty [X-CURSES]	scanw [LSB]

scr_dump [X-CURSES]	scr_init [X-CURSES]	scr_restore [X-CURSES]	scr_set [X-CURSES]
scrl [X-CURSES]	scroll [X-CURSES]	scrollok [X-CURSES]	set_curterm [X-CURSES]
set_term [X-CURSES]	setscrreg [X-CURSES]	setupterm [X-CURSES]	slk_attr_set [X-CURSES]
slk_attroff [X-CURSES]	slk_attron [X-CURSES]	slk_attrset [X-CURSES]	slk_clear [X-CURSES]
slk_color [X-CURSES]	slk_init [X-CURSES]	slk_label [X-CURSES]	slk_noutrefresh [X-CURSES]
slk_refresh [X-CURSES]	slk_restore [X-CURSES]	slk_set [X-CURSES]	slk_touch [X-CURSES]
standend [X-CURSES]	standout [X-CURSES]	start_color [X-CURSES]	subpad [X-CURSES]
subwin [X-CURSES]	syncok [X-CURSES]	termattrs [X-CURSES]	termname [X-CURSES]
tgetent [X-CURSES]	tgetflag [X-CURSES]	tgetnum [X-CURSES]	tgetstr [X-CURSES]
tgoto [X-CURSES]	tigetflag [X-CURSES]	tigetnum [X-CURSES]	tigetstr [X-CURSES]
timeout [X-CURSES]	touchline [X-CURSES]	touchwin [X-CURSES]	tparm [X-CURSES]
tputs [X-CURSES]	typeahead [X-CURSES]	unctrl [X-CURSES]	ungetch [X-CURSES]
untouchwin [X-CURSES]	use_env [X-CURSES]	vidattr [X-CURSES]	vidputs [X-CURSES]
vline [X-CURSES]	vw_printw [X-CURSES]	vw_scanw [LSB]	vwprintw [X-CURSES]
vwscanw [LSB]	waddch [X-CURSES]	waddchnstr [X-CURSES]	waddchstr [X-CURSES]
waddnstr [X-CURSES]	waddstr [X-CURSES]	wattr_get [X-CURSES]	wattr_off [X-CURSES]
wattr_on [X-CURSES]	wattr_set [X-CURSES]	wattroff [X-CURSES]	wattron [X-CURSES]
wattrset [X-CURSES]	wbkgd [X-CURSES]	wbkgdset [X-CURSES]	wborder [X-CURSES]
wchgat [X-CURSES]	wclear [X-CURSES]	wclrtobot [X-CURSES]	wclrtoeol [X-CURSES]

wcolor_set [X-CURSES]	wcursyncup [X-CURSES]	wdelch [X-CURSES]	wdeleteln [X-CURSES]
wechochar [X-CURSES]	werase [X-CURSES]	wgetch [X-CURSES]	wgetnstr [X-CURSES]
wgetstr [X-CURSES]	whline [X-CURSES]	winch [X-CURSES]	winchnstr [LSB]
winchstr [LSB]	winnstr [X-CURSES]	winsch [X-CURSES]	winsdelln [X-CURSES]
winsertln [X-CURSES]	winsnstr [X-CURSES]	winsstr [X-CURSES]	winstr [LSB]
wmove [X-CURSES]	wnoutrefresh [X-CURSES]	wprintw [X-CURSES]	wredrawln [X-CURSES]
wrefresh [X-CURSES]	wscanw [LSB]	wscrl [X-CURSES]	wsetscrreg [X-CURSES]
wstandend [X-CURSES]	wstandout [X-CURSES]	wsyncdown [X-CURSES]	wsyncup [X-CURSES]
wtimeout [X-CURSES]	wtouchln [X-CURSES]	wvline [X-CURSES]	

An LSB conforming implementation shall provide the generic deprecated functions for Curses specified in Table 15-5, 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 15-5 libncurses - Curses Deprecated Function Interfaces

tgetent [X-CURSES]	tgetflag [X-CURSES]	tgetnum [X-CURSES]	tgetstr [X-CURSES]
tgoto [X-CURSES]			

An LSB conforming implementation shall provide the generic data interfaces for Curses specified in Table 15-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 15-6 libncurses - Curses Data Interfaces

COLORS [X-CURSES]	COLOR_PAIRS [X-CURSES]	COLS [X-CURSES]	LINES [X-CURSES]
acs_map [X-CURSES]	cur_term [X-CURSES]	curscr [X-CURSES]	newscr [LSB]
stdscr [X-CURSES]	ttytype [X-CURSES]		

15.6 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.

15.6.1 curses.h

```
#define getattrs(win)   ((win)?(win)->_attrs:A_NORMAL)
#define ERR     (-1)
#define OK      (0)
#define ACS_RARROW      (acs_map['+'])
#define ACS_LARROW      (acs_map[','])
#define ACS_UARROW      (acs_map['-'])
#define ACS_DARROW      (acs_map['.'])
#define ACS_BLOCK       (acs_map['0'])
#define ACS_CKBOARD     (acs_map['a'])
#define ACS_DEGREE      (acs_map['f'])
#define ACS_PLMINUS     (acs_map['g'])
#define ACS_BOARD       (acs_map['h'])
#define ACS_LANTERN     (acs_map['i'])
#define ACS_LRCORNER    (acs_map['j'])
#define ACS_URCORNER    (acs_map['k'])
#define ACS_ULCORNER    (acs_map['l'])
#define ACS_LLCORNER    (acs_map['m'])
#define ACS_PLUS        (acs_map['n'])
#define ACS_S1  (acs_map['o'])
#define ACS_HLINE       (acs_map['q'])
#define ACS_S9  (acs_map['s'])
#define ACS_LTEE        (acs_map['t'])
#define ACS_RTEE        (acs_map['u'])
#define ACS_BTEE        (acs_map['v'])
#define ACS_TTEE        (acs_map['w'])
#define ACS_VLINE       (acs_map['x'])
#define ACS_DIAMOND     (acs_map['`'])
#define ACS_BULLET      (acs_map['~'])
#define   setsyx(y,x)                 do{if((y)==-1&&(x)==-1)newscr-
>_leaveok=TRUE;else{newscr-
>_leaveok=FALSE;wmove(newscr,(y),(x));}}while(0)
#define   getsyx(y,x)               do{if(newscr->_leaveok)(y)=(x)=-
1;elsegetyx(newscr,(y),(x));}while(0)
#define vid_attr(a,pair,opts)   vidattr(a)
#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;
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;               /* maximums of x and y, NOT window
size */
```

```
    short _maxx;
    short _begy;                /* screen coords of upper-left-hand
corner */
    short _begx;
    short _flags;              /* window state flags */
    attr_t _attrs;              /* current attribute for non-space
character */
    chtype _bkgd;              /* 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;
    short _yoffset;            /* 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
#define KEY_DL  0510
#define KEY_IL  0511
#define KEY_DC  0512
#define KEY_IC  0513
#define KEY_EIC 0514
#define KEY_CLEAR       0515
#define KEY_EOS 0516
#define KEY_EOL 0517
#define KEY_SF  0520
#define KEY_SR  0521
#define KEY_NPAGE       0522
#define KEY_PPAGE       0523
#define KEY_STAB        0524
#define KEY_CTAB        0525
#define KEY_CATAB       0526
```

```
#define KEY_ENTER      0527
#define KEY_SRESET     0530
#define KEY_RESET      0531
#define KEY_PRINT      0532
#define KEY_LL  0533
#define KEY_A1  0534
#define KEY_A3  0535
#define KEY_B2  0536
#define KEY_C1  0537
#define KEY_C3  0540
#define KEY_BTAB       0541
#define KEY_BEG 0542
#define KEY_CANCEL     0543
#define KEY_CLOSE      0544
#define KEY_COMMAND    0545
#define KEY_COPY       0546
#define KEY_CREATE     0547
#define KEY_END 0550
#define KEY_EXIT       0551
#define KEY_FIND       0552
#define KEY_HELP       0553
#define KEY_MARK       0554
#define KEY_MESSAGE    0555
#define KEY_MOVE       0556
#define KEY_NEXT       0557
#define KEY_OPEN       0560
#define KEY_OPTIONS    0561
#define KEY_PREVIOUS   0562
#define KEY_REDO       0563
#define KEY_REFERENCE  0564
#define KEY_REFRESH    0565
#define KEY_REPLACE    0566
#define KEY_RESTART    0567
#define KEY_RESUME     0570
#define KEY_SAVE       0571
#define KEY_SBEG       0572
#define KEY_SCANCEL    0573
#define KEY_SCOMMAND   0574
#define KEY_SCOPY      0575
#define KEY_SCREATE    0576
#define KEY_SDC 0577
#define KEY_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
#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_SRIGHT     0622
#define KEY_SRSUME     0623
#define KEY_SSAVE      0624
#define KEY_SSUSPEND   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 *, int);
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 attroff(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 box(WINDOW *, chtype, chtype);
extern bool can_change_color(void);
extern int cbreak(void);
extern int chgat(int, attr_t, short, const void *);
extern int clear(void);
extern int clearok(WINDOW *, bool);
extern int clrtobot(void);
extern int clrtoeol(void);
extern int color_content(short, short *, short *, short *);
extern int color_set(short, void *);
extern int copywin(const WINDOW *, WINDOW *, int, int, int, int,
int, int,
                   int);
extern int curs_set(int);
extern WINDOW *curscr;
```



```
extern int def_prog_mode(void);
extern int def_shell_mode(void);
extern int delay_output(int);
extern int delch(void);
extern int deleteln(void);
extern void delscreen(SCREEN *);
extern int delwin(WINDOW *);
extern WINDOW *derwin(WINDOW *, int, int, int, int);
extern int doupdate(void);
extern WINDOW *dupwin(WINDOW *);
extern int echo(void);
extern int echochar(const chtype);
extern int endwin(void);
extern int erase(void);
extern char erasechar(void);
extern void filter(void);
extern int flash(void);
extern int flushinp(void);
extern chtype getbkgd(WINDOW *);
extern int getch(void);
extern int getnstr(char *, int);
extern int getstr(char *);
extern WINDOW *getwin(FILE *);
extern int halfdelay(int);
extern bool has_colors(void);
extern bool has_ic(void);
extern bool has_il(void);
extern int hline(chtype, int);
extern void idcok(WINDOW *, bool);
extern int idlok(WINDOW *, bool);
extern void immedok(WINDOW *, bool);
extern chtype inch(void);
extern int inchnstr(chtype *, int);
extern int inchstr(chtype *);
extern int init_color(short, short, short, short);
extern int init_pair(short, short, short);
extern WINDOW *initscr(void);
extern int innstr(char *, int);
extern int insch(chtype);
extern int insdelln(int);
extern int insertln(void);
extern int insnstr(const char *, int);
extern int insstr(const char *);
extern int instr(char *);
extern int intrflush(WINDOW *, bool);
extern bool is_linetouched(WINDOW *, int);
extern bool is_wintouched(WINDOW *);
extern bool isendwin(void);
extern const char *keyname(int);
extern int keypad(WINDOW *, bool);
extern char killchar(void);
extern int leaveok(WINDOW *, bool);
extern char *longname(void);
extern int meta(WINDOW *, bool);
extern int move(int, int);
extern int mvaddch(int, int, const chtype);
extern int mvaddchnstr(int, int, const chtype *, int);
extern int mvaddchstr(int, int, const chtype *);
extern int mvaddnstr(int, int, const char *, int);
extern int mvaddstr(int, int, const char *);
extern int mvchgat(int, int, int, attr_t, short, const void *);
extern int mvcur(int, int, int, int);
extern int mvdelch(int, int);
extern int mvderwin(WINDOW *, int, int);
extern int mvgetch(int, int);
extern int mvgetnstr(int, int, char *, int);
```

```
extern int mvgetstr(int, int, char *);
extern 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 mvinstr(int, int, char *);
extern int mvprintw(int, int, const char *, ...);
extern int mvscanw(int, int, const char *, ...);
extern int mvvline(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 mvwchgat(WINDOW *, int, int, int, attr_t, short, const
void *);
extern int mvwdelch(WINDOW *, int, int);
extern int mvwgetch(WINDOW *, int, int);
extern int mvwgetnstr(WINDOW *, int, int, char *, int);
extern int mvwgetstr(WINDOW *, int, int, char *);
extern int mvwhline(WINDOW *, int, int, chtype, int);
extern int mvwin(WINDOW *, int, int);
extern chtype mvwinch(WINDOW *, int, int);
extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
extern int mvwinchstr(WINDOW *, int, int, chtype *);
extern int mvwinnstr(WINDOW *, int, int, char *, int);
extern int mvwinsch(WINDOW *, int, int, chtype);
extern int mvwinsnstr(WINDOW *, int, int, const char *, int);
extern int mvwinsstr(WINDOW *, int, int, const char *);
extern int mvwinstr(WINDOW *, int, int, char *);
extern int mvwprintw(WINDOW *, int, int, const char *, ...);
extern int mvwscanw(WINDOW *, int, int, const char *, ...);
extern int mvwvline(WINDOW *, int, int, chtype, int);
extern int napms(int);
extern WINDOW *newpad(int, int);
extern WINDOW *newscr;
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, int);
extern int prefresh(WINDOW *, int, int, int, int, int, int);
extern int printw(const char *, ...);
extern int putwin(WINDOW *, FILE *);
extern void qiflush(void);
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 resetty(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 scrl(int);
extern int scroll(WINDOW *);
extern int scrollok(WINDOW *, bool);
extern SCREEN *set_term(SCREEN *);
extern int setscrreg(int, int);
extern attr_t slk_attr(void);
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 char *slk_label(int);
extern int slk_noutrefresh(void);
extern int slk_refresh(void);
extern int slk_restore(void);
extern int slk_set(int, const char *, int);
extern int slk_touch(void);
extern int standend(void);
extern int standout(void);
extern int start_color(void);
extern WINDOW *stdscr;
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 vw_printw(WINDOW *, const char *, va_list);
extern int vw_scanw(WINDOW *, const char *, va_list);
extern int vwprintw(WINDOW *, const char *, va_list);
extern int vwscanw(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);
extern int waddstr(WINDOW *, const char *);
extern int wattr_get(WINDOW *, attr_t *, short *, void *);
extern int wattr_off(WINDOW *, attr_t, void *);
extern int wattr_on(WINDOW *, attr_t, void *);
extern int wattr_set(WINDOW *, attr_t, short, void *);
extern int wattroff(WINDOW *, int);
extern int wattron(WINDOW *, int);
extern int wattrset(WINDOW *, int);
extern int wbkgd(WINDOW *, chtype);
extern void wbkgdset(WINDOW *, chtype);
extern int wborder(WINDOW *, chtype, chtype, chtype, chtype, chtype,
                chtype, chtype, chtype);
extern int wchgat(WINDOW *, int, attr_t, short, const void *);
```

```
extern int wclear(WINDOW *);
extern int wclrtobot(WINDOW *);
extern int wclrtoeol(WINDOW *);
extern int wcolor_set(WINDOW *, short, void *);
extern void wcursyncup(WINDOW *);
extern int wdelch(WINDOW *);
extern int wdeleteln(WINDOW *);
extern int wechochar(WINDOW *, const chtype);
extern int werase(WINDOW *);
extern int wgetch(WINDOW *);
extern int wgetnstr(WINDOW *, char *, int);
extern int wgetstr(WINDOW *, char *);
extern int whline(WINDOW *, chtype, int);
extern chtype winch(WINDOW *);
extern int winchnstr(WINDOW *, chtype *, int);
extern int winchstr(WINDOW *, chtype *);
extern int winnstr(WINDOW *, char *, int);
extern int winsch(WINDOW *, chtype);
extern int winsdelln(WINDOW *, int);
extern int winsertln(WINDOW *);
extern int winsnstr(WINDOW *, const char *, int);
extern int winsstr(WINDOW *, const char *);
extern int winstr(WINDOW *, char *);
extern int wmove(WINDOW *, int, int);
extern int wnoutrefresh(WINDOW *);
extern int wprintw(WINDOW *, const char *, ...);
extern int wredrawln(WINDOW *, int, int);
extern int wrefresh(WINDOW *);
extern int wscanw(WINDOW *, const char *, ...);
extern int wscrl(WINDOW *, int);
extern int wsetscrreg(WINDOW *, int, int);
extern int wstandend(WINDOW *);
extern int wstandout(WINDOW *);
extern void wsyncdown(WINDOW *);
extern void wsyncup(WINDOW *);
extern void wtimeout(WINDOW *, int);
extern int wtouchln(WINDOW *, int, int, int);
extern int wvline(WINDOW *, chtype, int);
```

15.6.2 term.h

```
extern TERMINAL *cur_term;
extern int del_curterm(TERMINAL *);
extern int putp(const char *);
extern int restartterm(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 *);
extern char *tgetstr(char *, char **);
extern char *tgoto(const char *, int, int);
extern int tigetflag(const char *);
extern int tigetnum(const char *);
extern char *tigetstr(const char *);
extern char *tparm(const char *, ...);
extern int tputs(const char *, int, int (*)(int));
extern char ttytype[];
```

15.7 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 15.5 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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

```
#include <curses.h>
int mvinchstr(int y, int x, chtype * chstr);
```

Description

The interface `mvinchstr()` shall behave as specified in X/Open Curses, Issue 7, 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

```
#include <curses.h>
int mvinstr(int y, int x, char * str);
```

Description

The interface mvinstr() shall behave as specified in X/Open Curses, Issue 7, except that mvinstr() shall return the number of characters that were read.

mvscanw

Name

mvscanw — convert formatted input from a curses window

Synopsis

```
#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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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, Issue 7, 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 POSIX 1003.1-2008 (ISO/IEC 9945-2009), 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, Issue 7, 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, Issue 7, which indicates this function returns OK on success.

15.8 Interfaces for libncursesw

Table 15-7 defines the library name and shared object name for the libncursesw library

Table 15-7 libncursesw Definition

Library:	libncursesw
SONAME:	libncursesw.so.5

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

[Libncursesw] Libncursesw API
[LSB] This Specification
[ncursesw] Libncursesw Placeholder
[X-CURSES] X/Open Curses, Issue 7

15.8.1 Curses Wide

15.8.1.1 Interfaces for Curses Wide

An LSB conforming implementation shall provide the generic functions for Curses Wide specified in Table 15-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 15-8 libncursesw - Curses Wide Function Interfaces

add_wch [Libncursesw]	add_wchnstr [Libncursesw]	add_wchstr [Libncursesw]	addch [Libncursesw]
addchnstr [Libncursesw]	addchstr [Libncursesw]	addnstr [Libncursesw]	addnwstr [Libncursesw]

addstr [Libncursesw]	addwstr [Libncursesw]	assume_default_colors [Libncursesw]	attr_get [Libncursesw]
attr_off [Libncursesw]	attr_on [Libncursesw]	attr_set [Libncursesw]	attroff [Libncursesw]
attron [Libncursesw]	attrset [Libncursesw]	baudrate [Libncursesw]	beep [Libncursesw]
bkgd [Libncursesw]	bkgdset [Libncursesw]	bkgrnd [Libncursesw]	bkgrndset [Libncursesw]
border [Libncursesw]	border_set [Libncursesw]	box [Libncursesw]	box_set [Libncursesw]
can_change_color [Libncursesw]	cbreak [Libncursesw]	chgat [Libncursesw]	clear [Libncursesw]
clearok [Libncursesw]	clrtobot [Libncursesw]	clrtoeol [Libncursesw]	color_content [Libncursesw]
color_set [Libncursesw]	copywin [Libncursesw]	curs_set [Libncursesw]	curses_version [Libncursesw]
def_prog_mode [Libncursesw]	def_shell_mode [Libncursesw]	define_key [Libncursesw]	del_curterm [Libncursesw]
delay_output [Libncursesw]	delch [Libncursesw]	deleteln [Libncursesw]	delscreen [Libncursesw]
delwin [Libncursesw]	derwin [Libncursesw]	doupdate [Libncursesw]	dupwin [Libncursesw]
echo [Libncursesw]	echo_wchar [Libncursesw]	echochar [Libncursesw]	endwin [Libncursesw]
erase [Libncursesw]	erasechar [Libncursesw]	erasewchar [Libncursesw]	filter [Libncursesw]
flash [Libncursesw]	flushinp [Libncursesw]	get_wch [Libncursesw]	get_wstr [Libncursesw]
getbkgd [Libncursesw]	getbkgrnd [Libncursesw]	getcchar [Libncursesw]	getch [Libncursesw]
getmouse [Libncursesw]	getn_wstr [Libncursesw]	getnstr [Libncursesw]	getstr [Libncursesw]
getwin [Libncursesw]	halfdelay [Libncursesw]	has_colors [Libncursesw]	has_ic [Libncursesw]
has_il [Libncursesw]	has_key [Libncursesw]	has_mouse [LSB]	hline [Libncursesw]
hline_set [Libncursesw]	idcok [Libncursesw]	idlok [Libncursesw]	immedok [Libncursesw]

in_wch [Libncursesw]	in_wchnstr [Libncursesw]	in_wchstr [Libncursesw]	inch [Libncursesw]
inchnstr [Libncursesw]	inchstr [Libncursesw]	init_color [Libncursesw]	init_pair [Libncursesw]
initscr [Libncursesw]	innstr [Libncursesw]	innwstr [Libncursesw]	ins_nwstr [Libncursesw]
ins_wch [Libncursesw]	ins_wstr [Libncursesw]	insch [Libncursesw]	insdelln [Libncursesw]
insertln [Libncursesw]	insnstr [Libncursesw]	insstr [Libncursesw]	instr [Libncursesw]
intrflush [Libncursesw]	inwstr [Libncursesw]	is_linetouched [Libncursesw]	is_wintouched [Libncursesw]
isendwin [Libncursesw]	key_name [Libncursesw]	keybound [Libncursesw]	keyname [Libncursesw]
keyok [Libncursesw]	keypad [LSB]	killchar [Libncursesw]	killwchar [Libncursesw]
leaveok [Libncursesw]	longname [Libncursesw]	mcprint [Libncursesw]	meta [Libncursesw]
mouse_trafo [Libncursesw]	mouseinterval [Libncursesw]	mousemask [Libncursesw]	move [Libncursesw]
mvadd_wch [Libncursesw]	mvadd_wchnstr [Libncursesw]	mvadd_wchstr [Libncursesw]	mvaddch [Libncursesw]
mvaddchnstr [Libncursesw]	mvaddchstr [Libncursesw]	mvaddnstr [Libncursesw]	mvaddnwstr [Libncursesw]
mvaddstr [Libncursesw]	mvaddwstr [Libncursesw]	mvchgat [Libncursesw]	mvcur [Libncursesw]
mvdelch [Libncursesw]	mvderwin [Libncursesw]	mvget_wch [Libncursesw]	mvget_wstr [Libncursesw]
mvgetch [Libncursesw]	mvgetn_wstr [Libncursesw]	mvgetnstr [Libncursesw]	mvgetstr [Libncursesw]
mvhline [Libncursesw]	mvhline_set [Libncursesw]	mvin_wch [Libncursesw]	mvin_wchnstr [Libncursesw]
mvin_wchstr [Libncursesw]	mvinch [Libncursesw]	mvinchnstr [Libncursesw]	mvinchstr [Libncursesw]
mvinnstr [Libncursesw]	mvinnwstr [Libncursesw]	mvins_nwstr [Libncursesw]	mvins_wch [Libncursesw]
mvins_wstr [Libncursesw]	mvinsch [Libncursesw]	mvinsnstr [Libncursesw]	mvinsstr [Libncursesw]

mvinstr [Libncursesw]	mvinwstr [Libncursesw]	mvprintw [Libncursesw]	mvscanw [Libncursesw]
mvvline [Libncursesw]	mvvline_set [Libncursesw]	mvwadd_wch [Libncursesw]	mvwadd_wchnstr [Libncursesw]
mvwadd_wchstr [Libncursesw]	mvwaddch [Libncursesw]	mvwaddchnstr [Libncursesw]	mvwaddchstr [Libncursesw]
mvwaddnstr [Libncursesw]	mvwaddnwstr [Libncursesw]	mvwaddstr [Libncursesw]	mvwaddwstr [Libncursesw]
mvwchgat [Libncursesw]	mvwdelch [Libncursesw]	mvwget_wch [Libncursesw]	mvwget_wstr [Libncursesw]
mvwgetch [Libncursesw]	mvwgetn_wstr [Libncursesw]	mvwgetnstr [Libncursesw]	mvwgetstr [Libncursesw]
mvwhline [Libncursesw]	mvwhline_set [Libncursesw]	mvwin [Libncursesw]	mvwin_wch [Libncursesw]
mvwin_wchnstr [Libncursesw]	mvwin_wchstr [Libncursesw]	mvwinch [Libncursesw]	mvwinchnstr [Libncursesw]
mvwinchstr [Libncursesw]	mvwinnstr [Libncursesw]	mvwinnwstr [Libncursesw]	mvwins_nwstr [Libncursesw]
mvwins_wch [Libncursesw]	mvwins_wstr [Libncursesw]	mvwinsch [Libncursesw]	mvwinsnstr [Libncursesw]
mvwinsstr [Libncursesw]	mvwinstr [Libncursesw]	mvwinwstr [Libncursesw]	mvwprintw [Libncursesw]
mvwscanw [Libncursesw]	mvwvline [Libncursesw]	mvwvline_set [Libncursesw]	napms [Libncursesw]
newpad [Libncursesw]	newterm [Libncursesw]	newwin [Libncursesw]	nl [Libncursesw]
nocbreak [Libncursesw]	nodelay [Libncursesw]	noecho [Libncursesw]	nonl [Libncursesw]
noqiflush [Libncursesw]	noraw [Libncursesw]	notimeout [Libncursesw]	overlay [Libncursesw]
overwrite [Libncursesw]	pair_content [Libncursesw]	pecho_wchar [Libncursesw]	pechochar [Libncursesw]
pnoutrefresh [Libncursesw]	prefresh [Libncursesw]	printw [Libncursesw]	putp [Libncursesw]
putwin [Libncursesw]	qiflush [Libncursesw]	raw [Libncursesw]	redrawwin [Libncursesw]
refresh [Libncursesw]	reset_prog_mode [Libncursesw]	reset_shell_mode [Libncursesw]	resetty [Libncursesw]

resizeterm [Libncursesw]	restartterm [Libncursesw]	ripoffline [Libncursesw]	savetty [Libncursesw]
scanw [Libncursesw]	scr_dump [Libncursesw]	scr_init [Libncursesw]	scr_restore [Libncursesw]
scr_set [Libncursesw]	scrl [Libncursesw]	scroll [Libncursesw]	scrollok [Libncursesw]
set_curterm [Libncursesw]	set_term [Libncursesw]	setcchar [Libncursesw]	setscrreg [Libncursesw]
setupterm [Libncursesw]	slk_attr [X-CURSES]	slk_attr_off [X-CURSES]	slk_attr_on [X-CURSES]
slk_attr_set [Libncursesw]	slk_attroff [Libncursesw]	slk_attron [Libncursesw]	slk_attrset [Libncursesw]
slk_clear [Libncursesw]	slk_color [Libncursesw]	slk_init [Libncursesw]	slk_label [Libncursesw]
slk_noutrefresh [Libncursesw]	slk_refresh [Libncursesw]	slk_restore [Libncursesw]	slk_set [Libncursesw]
slk_touch [Libncursesw]	slk_wset [Libncursesw]	standend [Libncursesw]	standout [Libncursesw]
start_color [Libncursesw]	subpad [Libncursesw]	subwin [Libncursesw]	syncok [Libncursesw]
term_attrs [X-CURSES]	termattrs [Libncursesw]	termname [Libncursesw]	tgetent [Libncursesw]
tgetflag [Libncursesw]	tgetnum [Libncursesw]	tgetstr [Libncursesw]	tgoto [Libncursesw]
tigetflag [Libncursesw]	tigetnum [Libncursesw]	tigetstr [Libncursesw]	timeout [Libncursesw]
touchline [Libncursesw]	touchwin [Libncursesw]	tparm [Libncursesw]	tputs [Libncursesw]
typeahead [Libncursesw]	unctrl [Libncursesw]	unget_wch [Libncursesw]	ungetch [Libncursesw]
ungetmouse [Libncursesw]	untouchwin [Libncursesw]	use_default_colors [Libncursesw]	use_env [Libncursesw]
use_extended_names [Libncursesw]	vid_attr [X-CURSES]	vid_puts [Libncursesw]	vidattr [Libncursesw]
vidputs [Libncursesw]	vline [Libncursesw]	vline_set [Libncursesw]	vw_printw [Libncursesw]
vw_scanw [Libncursesw]	vwprintw [Libncursesw]	vwscanw [Libncursesw]	wadd_wch [Libncursesw]

wadd_wchnstr [Libncursesw]	wadd_wchstr [Libncursesw]	waddch [Libncursesw]	waddchnstr [Libncursesw]
waddchstr [Libncursesw]	waddnstr [Libncursesw]	waddnwstr [Libncursesw]	waddstr [Libncursesw]
waddwstr [Libncursesw]	wattr_get [Libncursesw]	wattr_off [Libncursesw]	wattr_on [Libncursesw]
wattr_set [Libncursesw]	wattroff [Libncursesw]	wattron [Libncursesw]	wattrset [Libncursesw]
wbkgd [Libncursesw]	wbkgdset [Libncursesw]	wbkgrnd [Libncursesw]	wbkgrndset [Libncursesw]
wborder [Libncursesw]	wborder_set [Libncursesw]	wchgat [Libncursesw]	wclear [Libncursesw]
wclrtobot [Libncursesw]	wclrtoeol [Libncursesw]	wcolor_set [Libncursesw]	wcursyncup [Libncursesw]
wdelch [Libncursesw]	wdeleteln [Libncursesw]	wecho_wchar [Libncursesw]	wechochar [Libncursesw]
werase [Libncursesw]	wget_wch [Libncursesw]	wget_wstr [Libncursesw]	wgetbkgrnd [Libncursesw]
wgetch [Libncursesw]	wgetn_wstr [Libncursesw]	wgetnstr [Libncursesw]	wgetstr [Libncursesw]
whline [Libncursesw]	whline_set [Libncursesw]	win_wch [Libncursesw]	win_wchnstr [Libncursesw]
win_wchstr [Libncursesw]	winch [Libncursesw]	winchnstr [Libncursesw]	winchstr [Libncursesw]
winnstr [Libncursesw]	winnwstr [Libncursesw]	wins_nwstr [Libncursesw]	wins_wch [Libncursesw]
wins_wstr [Libncursesw]	winsch [Libncursesw]	winsdelln [Libncursesw]	winsertln [Libncursesw]
winsnstr [Libncursesw]	winsstr [Libncursesw]	winstr [Libncursesw]	winwstr [Libncursesw]
wmouse_trafo [Libncursesw]	wmove [Libncursesw]	wnoutrefresh [Libncursesw]	wprintw [Libncursesw]
wredrawln [Libncursesw]	wrefresh [Libncursesw]	wresize [Libncursesw]	wscanw [Libncursesw]
wscrl [Libncursesw]	wsetscrreg [Libncursesw]	wstandend [Libncursesw]	wstandout [Libncursesw]
wsyncdown [Libncursesw]	wsyncup [Libncursesw]	wtimeout [Libncursesw]	wtouchln [Libncursesw]

wunctrl [Libncursesw]	wvline [Libncursesw]	wvline_set [Libncursesw]	

An LSB conforming implementation shall provide the generic deprecated functions for Curses Wide specified in Table 15-9, 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 15-9 libncursesw - Curses Wide Deprecated Function Interfaces

tgetent [Libncursesw]	tgetflag [Libncursesw]	tgetnum [Libncursesw]	tgetstr [Libncursesw]
tgoto [Libncursesw]			

An LSB conforming implementation shall provide the generic data interfaces for Curses Wide specified in Table 15-10, with the full mandatory functionality as described in the referenced underlying specification.

Table 15-10 libncursesw - Curses Wide Data Interfaces

COLORS [ncursesw]	COLOR_PAIRS [ncursesw]	COLS [ncursesw]	LINES [ncursesw]
acs_map [LSB]	cur_term [LSB]	curscr [ncursesw]	newscr [ncursesw]
stdscr [ncursesw]	ttytype [ncursesw]		

15.9 Data Definitions for libncursesw

This section defines global identifiers and their values that are associated with interfaces contained in libncursesw. 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.

15.9.1 ncursesw/curses.h

```
#define CURSES  1
#define setsyx(y,x)     do { if (newscr) { \
                             if ((y) == -1 && (x) == -1) \
                                    leaveok(newscr, TRUE); \
                             else { \
                                    leaveok(newscr, FALSE); \
                                    wmove(newscr, (y), (x)); \
                                  } \
                           } \
                         } while(0)
#define getsyx(y,x)     do { if (newscr) { \
                             if (is_leaveok(newscr)) \
                                    (y) = (x) = -1; \
                             else \
                                    getyx(newscr,(y), (x)); \
                           } \
                        } while(0)
#define CURSES_H       1
#define NCURSES_VERSION_MAJOR   5
#define NCURSES_VERSION_MINOR   9
#define NCURSES_VERSION_PATCH   20110404
#define NCURSES_VERSION "5.9"
#define NCURSES_MOUSE_VERSION   1
#define NCURSES_ENABLE_STDBOOL_H       1
#define NCURSES_INLINE  inline
#define NCURSES_TPARM_VARARGS   1
#ifndef TRUE
#define TRUE    1
#endif
#define NCURSES_BOOL    bool
#ifdef __cplusplus
#  define NCURSES_CAST(type,value)          static_cast<type>(value)
#else
#  define NCURSES_CAST(type,value)          (type)(value)
#endif
#define WA_ATTRIBUTES   A_ATTRIBUTES
#define WA_NORMAL       A_NORMAL
#define WA_STANDOUT     A_STANDOUT
#define WA_UNDERLINE    A_UNDERLINE
#define WA_REVERSE      A_REVERSE
#define WA_BLINK        A_BLINK
#define WA_DIM  A_DIM
#define WA_BOLD A_BOLD
#define WA_ALTCHARSET   A_ALTCHARSET
#define WA_INVIS        A_INVIS
#define WA_PROTECT      A_PROTECT
#define WA_HORIZONTAL   A_HORIZONTAL
#define WA_LEFT A_LEFT
#define WA_LOW  A_LOW
#define WA_RIGHT        A_RIGHT
#define WA_TOP  A_TOP
#define WA_VERTICAL     A_VERTICAL
#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 NCURSES_ACS(c)  (acs_map[NCURSES_CAST(unsigned char,c)])
#define ACS_ULCORNER    NCURSES_ACS('l')
#define ACS_LLCORNER    NCURSES_ACS('m')
#define ACS_URCORNER    NCURSES_ACS('k')
#define ACS_LRCORNER    NCURSES_ACS('j')
#define ACS_LTEE        NCURSES_ACS('t')
```

```
#define ACS_RTEE       NCURSES_ACS('u')
#define ACS_BTEE       NCURSES_ACS('v')
#define ACS_TTEE       NCURSES_ACS('w')
#define ACS_HLINE      NCURSES_ACS('q')
#define ACS_VLINE      NCURSES_ACS('x')
#define ACS_PLUS       NCURSES_ACS('n')
#define ACS_S1  NCURSES_ACS('o')
#define ACS_S9  NCURSES_ACS('s')
#define ACS_DIAMOND    NCURSES_ACS('`')
#define ACS_CKBOARD    NCURSES_ACS('a')
#define ACS_DEGREE     NCURSES_ACS('f')
#define ACS_PLMINUS    NCURSES_ACS('g')
#define ACS_BULLET     NCURSES_ACS('~')
#define ACS_LARROW     NCURSES_ACS(',')
#define ACS_RARROW     NCURSES_ACS('+')
#define ACS_DARROW     NCURSES_ACS('.')
#define ACS_UARROW     NCURSES_ACS('-')
#define ACS_BOARD      NCURSES_ACS('h')
#define ACS_LANTERN    NCURSES_ACS('i')
#define ACS_BLOCK      NCURSES_ACS('0')
#define ACS_S3  NCURSES_ACS('p')
#define ACS_S7  NCURSES_ACS('r')
#define ACS_LEQUAL     NCURSES_ACS('y')
#define ACS_GEQUAL     NCURSES_ACS('z')
#define ACS_PI  NCURSES_ACS('{')
#define ACS_NEQUAL     NCURSES_ACS('|')
#define ACS_STERLING   NCURSES_ACS('}')
#define ACS_BSSB       ACS_ULCORNER
#define ACS_SSBB       ACS_LLCORNER
#define ACS_BBSS       ACS_URCORNER
#define ACS_SBBS       ACS_LRCORNER
#define ACS_SBSS       ACS_RTEE
#define ACS_SSSB       ACS_LTEE
#define ACS_SSBS       ACS_BTEE
#define ACS_BSSS       ACS_TTEE
#define ACS_BSBS       ACS_HLINE
#define ACS_SBSB       ACS_VLINE
#define ACS_SSSS       ACS_PLUS
#define ERR    (-1)
#define OK     (0)
#define _SUBWIN 0x01
#define _ENDLINE       0x02
#define _FULLWIN       0x04
#define _SCROLLWIN     0x08
#define _ISPAD  0x10
#define _HASMOVED      0x20
#define _WRAPPED       0x40
#define _NOCHANGE      -1
#define _NEWINDEX      -1
#define CCHARW_MAX     5
#define NCURSES_EXT_COLORS     20110404
#define                                        GCC_PRINTFLIKE(fmt,var)
__attribute__((format(printf,fmt,var)))
#define                                         GCC_SCANFLIKE(fmt,var)
__attribute__((format(scanf,fmt,var)))
#define NCURSES_EXT_FUNCS      20110404
#define curses_version()       NCURSES_VERSION
#define NCURSES_SP_FUNCS       20110404
#define NCURSES_SP_OUTC NCURSES_SP_NAME(NCURSES_OUTC)
#define NCURSES_SP_NAME(name)   name
#define NCURSES_ATTR_SHIFT     8
#define  NCURSES_BITS(mask,shift)            ((mask) << ((shift) +
NCURSES_ATTR_SHIFT))
#define A_NORMAL       (1UL - 1UL)
#define A_ATTRIBUTES    NCURSES_BITS(~(1UL - 1UL),0)
#define A_CHARTEXT     (NCURSES_BITS(1UL,0) - 1UL)
```



```
#define A_COLOR NCURSES_BITS(((1UL) << 8) - 1UL,0)
#define A_STANDOUT     NCURSES_BITS(1UL,8)
#define A_UNDERLINE    NCURSES_BITS(1UL,9)
#define A_REVERSE      NCURSES_BITS(1UL,10)
#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 getyx(win,y,x)  (y = getcury(win), x = getcurx(win))
#define getbegyx(win,y,x)       (y = getbegy(win), x = getbegx(win))
#define getmaxyx(win,y,x)       (y = getmaxy(win), x = getmaxx(win))
#define getparyx(win,y,x)       (y = getpary(win), x = getparx(win))
#define wgetstr(w, s)   wgetnstr(w, s, -1)
#define getnstr(s, n)   wgetnstr(stdscr, s, n)
#define setterm(term)   setupterm(term, 1, (int *)0)
#define fixterm()       reset_prog_mode()
#define resetterm()     reset_shell_mode()
#define saveterm()      def_prog_mode()
#define crmode()       cbreak()
#define nocrmode()      nocbreak()
#define getattrs(win)    NCURSES_CAST(int, (win) ? (win)->_attrs :
A_NORMAL)
#define getcurx(win)    ((win) ? (win)->_curx : ERR)
#define getcury(win)    ((win) ? (win)->_cury : ERR)
#define getbegx(win)    ((win) ? (win)->_begx : ERR)
#define getbegy(win)    ((win) ? (win)->_begy : ERR)
#define getmaxx(win)    ((win) ? ((win)->_maxx + 1) : ERR)
#define getmaxy(win)    ((win) ? ((win)->_maxy + 1) : ERR)
#define getparx(win)    ((win) ? (win)->_parx : ERR)
#define getpary(win)    ((win) ? (win)->_pary : ERR)
#define wstandout(win)  (wattrset(win,A_STANDOUT))
#define wstandend(win)  (wattrset(win,A_NORMAL))
#define  wattron(win,at)  wattr_on(win,  NCURSES_CAST(attr_t,  at),
NULL)
#define wattroff(win,at)       wattr_off(win, NCURSES_CAST(attr_t,
at), NULL)
#define scroll(win)     wscrl(win,1)
#define touchwin(win)   wtouchln((win), 0, getmaxy(win), 1)
#define touchline(win, s, c)    wtouchln((win), s, c, 1)
#define untouchwin(win) wtouchln((win), 0, getmaxy(win), 0)
#define box(win, v, h)  wborder(win, v, v, h, h, 0, 0, 0, 0)
#define border(ls, rs, ts, bs, tl, tr, bl, br)  wborder(stdscr, ls,
rs, ts, bs, tl, tr, bl, br)
#define hline(ch, n)    whline(stdscr, ch, n)
#define vline(ch, n)    wvline(stdscr, ch, n)
#define winstr(w, s)    winnstr(w, s, -1)
#define winchstr(w, s)  winchnstr(w, s, -1)
#define winsstr(w, s)   winsnstr(w, s, -1)
#define redrawwin(win)  wredrawln(win, 0, (win)->_maxy+1)
#define waddstr(win,str)       waddnstr(win,str,-1)
#define waddchstr(win,str)     waddchnstr(win,str,-1)
#define COLOR_PAIR(n)   NCURSES_BITS(n, 0)
#define PAIR_NUMBER(a)   (NCURSES_CAST(int,((NCURSES_CAST(unsigned
long,a) & A_COLOR) >> NCURSES_ATTR_SHIFT)))
#define addch(ch)      waddch(stdscr,ch)
#define addchnstr(str,n)       waddchnstr(stdscr,str,n)
#define addchstr(str)   waddchstr(stdscr,str)
#define addnstr(str,n)  waddnstr(stdscr,str,n)
```

```
#define addstr(str)     waddnstr(stdscr,str,-1)
#define attroff(at)     wattroff(stdscr,at)
#define attron(at)      wattron(stdscr,at)
#define attrset(at)     wattrset(stdscr,at)
#define attr_get(ap,cp,o)       wattr_get(stdscr,ap,cp,o)
#define attr_off(a,o)   wattr_off(stdscr,a,o)
#define attr_on(a,o)    wattr_on(stdscr,a,o)
#define attr_set(a,c,o) wattr_set(stdscr,a,c,o)
#define bkgd(ch)        wbkgd(stdscr,ch)
#define bkgdset(ch)     wbkgdset(stdscr,ch)
#define chgat(n,a,c,o)  wchgat(stdscr,n,a,c,o)
#define clear() wclear(stdscr)
#define clrtobot()      wclrtobot(stdscr)
#define clrtoeol()      wclrtoeol(stdscr)
#define color_set(c,o)  wcolor_set(stdscr,c,o)
#define delch() wdelch(stdscr)
#define deleteln()      winsdelln(stdscr,-1)
#define echochar(c)     wechochar(stdscr,c)
#define erase() werase(stdscr)
#define getch() wgetch(stdscr)
#define getstr(str)     wgetstr(stdscr,str)
#define inch()  winch(stdscr)
#define inchnstr(s,n)   winchnstr(stdscr,s,n)
#define inchstr(s)      winchstr(stdscr,s)
#define innstr(s,n)     winnstr(stdscr,s,n)
#define insch(c)        winsch(stdscr,c)
#define insdelln(n)     winsdelln(stdscr,n)
#define insertln()      winsdelln(stdscr,1)
#define insnstr(s,n)    winsnstr(stdscr,s,n)
#define insstr(s)       winsstr(stdscr,s)
#define instr(s)        winstr(stdscr,s)
#define move(y,x)       wmove(stdscr,y,x)
#define refresh()       wrefresh(stdscr)
#define scrl(n) wscrl(stdscr,n)
#define setscrreg(t,b)  wsetscrreg(stdscr,t,b)
#define standend()      wstandend(stdscr)
#define standout()      wstandout(stdscr)
#define timeout(delay)  wtimeout(stdscr,delay)
#define wdeleteln(win)  winsdelln(win,-1)
#define winsertln(win)  winsdelln(win,1)
#define  mvwaddch(win,y,x,ch)       (wmove(win,y,x)  ==  ERR  ?  ERR  :
waddch(win,ch))
#define  mvwaddchnstr(win,y,x,str,n)       (wmove(win,y,x)  ==  ERR  ?
ERR : waddchnstr(win,str,n))
#define  mvwaddchstr(win,y,x,str)          (wmove(win,y,x)  ==  ERR  ?
ERR : waddchnstr(win,str,-1))
#define  mvwaddnstr(win,y,x,str,n)         (wmove(win,y,x)  ==  ERR  ?
ERR : waddnstr(win,str,n))
#define  mvwaddstr(win,y,x,str)    (wmove(win,y,x)  ==  ERR  ?  ERR  :
waddnstr(win,str,-1))
#define  mvwdelch(win,y,x)          (wmove(win,y,x)  ==  ERR  ?  ERR  :
wdelch(win))
#define mvwchgat(win,y,x,n,a,c,o)         (wmove(win,y,x)  ==  ERR  ?
ERR : wchgat(win,n,a,c,o))
#define  mvwgetch(win,y,x)          (wmove(win,y,x)  ==  ERR  ?  ERR  :
wgetch(win))
#define mvwgetnstr(win,y,x,str,n)         (wmove(win,y,x)  ==  ERR  ?
ERR : wgetnstr(win,str,n))
#define  mvwgetstr(win,y,x,str)    (wmove(win,y,x)  ==  ERR  ?  ERR  :
wgetstr(win,str))
#define  mvwhline(win,y,x,c,n)     (wmove(win,y,x)  ==  ERR  ?  ERR  :
whline(win,c,n))
#define  mvwinch(win,y,x)               (wmove(win,y,x)  ==  ERR  ?
NCURSES_CAST(chtype, ERR) : winch(win))
#define mvwinchnstr(win,y,x,s,n)          (wmove(win,y,x)  ==  ERR  ?
ERR : winchnstr(win,s,n))
```

```
#define mvwinchstr(win,y,x,s)     (wmove(win,y,x) == ERR ? ERR :
winchstr(win,s))
#define mvwinnstr(win,y,x,s,n)    (wmove(win,y,x) == ERR ? ERR :
winnstr(win,s,n))
#define mvwinsch(win,y,x,c)       (wmove(win,y,x) == ERR ? ERR :
winsch(win,c))
#define mvwinsnstr(win,y,x,s,n)  (wmove(win,y,x) == ERR ? ERR :
winsnstr(win,s,n))
#define mvwinsstr(win,y,x,s)      (wmove(win,y,x) == ERR ? ERR :
winsstr(win,s))
#define mvwinstr(win,y,x,s)       (wmove(win,y,x) == ERR ? ERR :
winstr(win,s))
#define mvwvline(win,y,x,c,n)     (wmove(win,y,x) == ERR ? ERR :
wvline(win,c,n))
#define mvaddch(y,x,ch) mvwaddch(stdscr,y,x,ch)
#define mvaddchnstr(y,x,str,n)  mvwaddchnstr(stdscr,y,x,str,n)
#define mvaddchstr(y,x,str)     mvwaddchstr(stdscr,y,x,str)
#define mvaddnstr(y,x,str,n)    mvwaddnstr(stdscr,y,x,str,n)
#define mvaddstr(y,x,str)       mvwaddstr(stdscr,y,x,str)
#define mvchgat(y,x,n,a,c,o)    mvwchgat(stdscr,y,x,n,a,c,o)
#define mvdelch(y,x)    mvwdelch(stdscr,y,x)
#define mvgetch(y,x)    mvwgetch(stdscr,y,x)
#define mvgetnstr(y,x,str,n)    mvwgetnstr(stdscr,y,x,str,n)
#define mvgetstr(y,x,str)       mvwgetstr(stdscr,y,x,str)
#define mvhline(y,x,c,n)        mvwhline(stdscr,y,x,c,n)
#define mvinch(y,x)     mvwinch(stdscr,y,x)
#define mvinchnstr(y,x,s,n)     mvwinchnstr(stdscr,y,x,s,n)
#define mvinchstr(y,x,s)        mvwinchstr(stdscr,y,x,s)
#define mvinnstr(y,x,s,n)       mvwinnstr(stdscr,y,x,s,n)
#define mvinsch(y,x,c)  mvwinsch(stdscr,y,x,c)
#define mvinsnstr(y,x,s,n)      mvwinsnstr(stdscr,y,x,s,n)
#define mvinsstr(y,x,s) mvwinsstr(stdscr,y,x,s)
#define mvinstr(y,x,s)  mvwinstr(stdscr,y,x,s)
#define mvvline(y,x,c,n)        mvwvline(stdscr,y,x,c,n)
#define getbkgd(win)    ((win)->_bkgd)
#define slk_attr_off(a,v)       ((v) ? ERR : slk_attroff(a))
#define slk_attr_on(a,v)        ((v) ? ERR : slk_attron(a))
#define wattr_set(win,a,p,opts) ((win)->_attrs = (((a) & ~A_COLOR)
| (attr_t)COLOR_PAIR(p)), OK)
#define vw_printw       vwprintw
#define vw_scanw        vwscanw
#define vsscanf(a,b,c)  _nc_vsscanf(a,b,c)
#define is_cleared(win) ((win) ? (win)->_clear : FALSE)
#define is_idcok(win)   ((win) ? (win)->_idcok : FALSE)
#define is_idlok(win)   ((win) ? (win)->_idlok : FALSE)
#define is_immedok(win) ((win) ? (win)->_immed : FALSE)
#define is_keypad(win)  ((win) ? (win)->_use_keypad : FALSE)
#define is_leaveok(win) ((win) ? (win)->_leaveok : FALSE)
#define is_nodelay(win) ((win) ? ((win)->_delay == 0) : FALSE)
#define is_notimeout(win)       ((win) ? (win)->_notimeout : FALSE)
#define is_pad(win)     ((win) ? ((win)->_flags & _ISPAD) != 0 :
FALSE)
#define is_scrollok(win)        ((win) ? (win)->_scroll : FALSE)
#define is_subwin(win)  ((win) ? ((win)->_flags & _SUBWIN) != 0 :
FALSE)
#define is_syncok(win)  ((win) ? (win)->_sync : FALSE)
#define wgetparent(win) ((win) ? (win)->_parent : 0)
#define wgetscrreg(win,t,b)     ((win) ? (*(t) = (win)->_regtop,
*(b) = (win)->_regbottom, OK) : ERR)
#define KEY_CODE_YES    0400
#define KEY_MIN 0401
#define KEY_BREAK       0401
#define KEY_SRESET      0530
#define KEY_RESET       0531
#define KEY_DOWN        0402
#define KEY_UP  0403
```

```
#define KEY_LEFT       0404
#define KEY_RIGHT      0405
#define KEY_HOME       0406
#define KEY_BACKSPACE   0407
#define KEY_F0  0410
#define KEY_F(n)       (KEY_F0+(n))
#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_PRINT      0532
#define KEY_LL  0533
#define KEY_A1  0534
#define KEY_A3  0535
#define KEY_B2  0536
#define KEY_C1  0537
#define KEY_C3  0540
#define KEY_BTAB       0541
#define KEY_BEG 0542
#define KEY_CANCEL     0543
#define KEY_CLOSE      0544
#define KEY_COMMAND    0545
#define KEY_COPY       0546
#define KEY_CREATE     0547
#define KEY_END 0550
#define KEY_EXIT       0551
#define KEY_FIND       0552
#define KEY_HELP       0553
#define KEY_MARK       0554
#define KEY_MESSAGE    0555
#define KEY_MOVE       0556
#define KEY_NEXT       0557
#define KEY_OPEN       0560
#define KEY_OPTIONS    0561
#define KEY_PREVIOUS   0562
#define KEY_REDO       0563
#define KEY_REFERENCE   0564
#define KEY_REFRESH    0565
#define KEY_REPLACE    0566
#define KEY_RESTART    0567
#define KEY_RESUME     0570
#define KEY_SAVE       0571
#define KEY_SBEG       0572
#define KEY_SCANCEL    0573
#define KEY_SCOMMAND   0574
#define KEY_SCOPY      0575
#define KEY_SCREATE    0576
#define KEY_SDC 0577
#define KEY_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
#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_SRIGHT     0622
#define KEY_SRSUME     0623
#define KEY_SSAVE      0624
#define KEY_SSUSPEND    0625
#define KEY_SUNDO      0626
#define KEY_SUSPEND    0627
#define KEY_UNDO       0630
#define KEY_MOUSE      0631
#define KEY_RESIZE     0632
#define KEY_EVENT      0633
#define KEY_MAX 0777
#define _XOPEN_CURSES   1
#define NCURSES_WACS(c) (&_nc_wacs[(unsigned char)c])
#define WACS_BSSB      NCURSES_WACS('l')
#define WACS_SSBB      NCURSES_WACS('m')
#define WACS_BBSS      NCURSES_WACS('k')
#define WACS_SBBS      NCURSES_WACS('j')
#define WACS_SBSS      NCURSES_WACS('u')
#define WACS_SSSB      NCURSES_WACS('t')
#define WACS_SSBS      NCURSES_WACS('v')
#define WACS_BSSS      NCURSES_WACS('w')
#define WACS_BSBS      NCURSES_WACS('q')
#define WACS_SBSB      NCURSES_WACS('x')
#define WACS_SSSS      NCURSES_WACS('n')
#define WACS_ULCORNER   WACS_BSSB
#define WACS_LLCORNER   WACS_SSBB
#define WACS_URCORNER   WACS_BBSS
#define WACS_LRCORNER   WACS_SBBS
#define WACS_RTEE      WACS_SBSS
#define WACS_LTEE      WACS_SSSB
#define WACS_BTEE      WACS_SSBS
#define WACS_TTEE      WACS_BSSS
#define WACS_HLINE     WACS_BSBS
#define WACS_VLINE     WACS_SBSB
#define WACS_PLUS      WACS_SSSS
#define WACS_S1 NCURSES_WACS('o')
#define WACS_S9 NCURSES_WACS('s')
#define WACS_DIAMOND    NCURSES_WACS('`')
#define WACS_CKBOARD    NCURSES_WACS('a')
#define WACS_DEGREE     NCURSES_WACS('f')
#define WACS_PLMINUS    NCURSES_WACS('g')
#define WACS_BULLET     NCURSES_WACS('~')
#define WACS_LARROW     NCURSES_WACS(',')
#define WACS_RARROW     NCURSES_WACS('+')
#define WACS_DARROW     NCURSES_WACS('.')
#define WACS_UARROW     NCURSES_WACS('-')
#define WACS_BOARD      NCURSES_WACS('h')
#define WACS_LANTERN    NCURSES_WACS('i')
#define WACS_BLOCK      NCURSES_WACS('0')
#define WACS_S3 NCURSES_WACS('p')
#define WACS_S7 NCURSES_WACS('r')
#define WACS_LEQUAL     NCURSES_WACS('y')
#define WACS_GEQUAL     NCURSES_WACS('z')
#define WACS_PI NCURSES_WACS('{')
```

```
#define WACS_NEQUAL     NCURSES_WACS('|')
#define WACS_STERLING   NCURSES_WACS('}')
#define WACS_BDDB       NCURSES_WACS('C')
#define WACS_DDBB       NCURSES_WACS('D')
#define WACS_BBDD       NCURSES_WACS('B')
#define WACS_DBBD       NCURSES_WACS('A')
#define WACS_DBDD       NCURSES_WACS('G')
#define WACS_DDDB       NCURSES_WACS('F')
#define WACS_DDBD       NCURSES_WACS('H')
#define WACS_BDDD       NCURSES_WACS('I')
#define WACS_BDBD       NCURSES_WACS('R')
#define WACS_DBDB       NCURSES_WACS('Y')
#define WACS_DDDD       NCURSES_WACS('E')
#define WACS_D_ULCORNER WACS_BDDB
#define WACS_D_LLCORNER WACS_DDBB
#define WACS_D_URCORNER WACS_BBDD
#define WACS_D_LRCORNER WACS_DBBD
#define WACS_D_RTEE     WACS_DBDD
#define WACS_D_LTEE     WACS_DDDB
#define WACS_D_BTEE     WACS_DDBD
#define WACS_D_TTEE     WACS_BDDD
#define WACS_D_HLINE    WACS_BDBD
#define WACS_D_VLINE    WACS_DBDB
#define WACS_D_PLUS     WACS_DDDD
#define WACS_BTTB       NCURSES_WACS('L')
#define WACS_TTBB       NCURSES_WACS('M')
#define WACS_BBTT       NCURSES_WACS('K')
#define WACS_TBBT       NCURSES_WACS('J')
#define WACS_TBTT       NCURSES_WACS('U')
#define WACS_TTTB       NCURSES_WACS('T')
#define WACS_TTBT       NCURSES_WACS('V')
#define WACS_BTTT       NCURSES_WACS('W')
#define WACS_BTBT       NCURSES_WACS('Q')
#define WACS_TBTB       NCURSES_WACS('X')
#define WACS_TTTT       NCURSES_WACS('N')
#define WACS_T_ULCORNER WACS_BTTB
#define WACS_T_LLCORNER WACS_TTBB
#define WACS_T_URCORNER WACS_BBTT
#define WACS_T_LRCORNER WACS_TBBT
#define WACS_T_RTEE     WACS_TBTT
#define WACS_T_LTEE     WACS_TTTB
#define WACS_T_BTEE     WACS_TTBT
#define WACS_T_TTEE     WACS_BTTT
#define WACS_T_HLINE    WACS_BTBT
#define WACS_T_VLINE    WACS_TBTB
#define WACS_T_PLUS     WACS_TTTT
#define add_wch(c)      wadd_wch(stdscr,c)
#define add_wchnstr(str,n)      wadd_wchnstr(stdscr,str,n)
#define add_wchstr(str) wadd_wchstr(stdscr,str)
#define addnwstr(wstr,n)        waddnwstr(stdscr,wstr,n)
#define addwstr(wstr)   waddwstr(stdscr,wstr)
#define bkgrnd(c)       wbkgrnd(stdscr,c)
#define bkgrndset(c)    wbkgrndset(stdscr,c)
#define                                 border_set(l,r,t,b,tl,tr,bl,br)
wborder_set(stdscr,l,r,t,b,tl,tr,bl,br)
#define box_set(w,v,h)  wborder_set(w,v,v,h,h,0,0,0,0)
#define echo_wchar(c)   wecho_wchar(stdscr,c)
#define get_wch(c)      wget_wch(stdscr,c)
#define get_wstr(t)     wget_wstr(stdscr,t)
#define getbkgrnd(wch)  wgetbkgrnd(stdscr,wch)
#define getn_wstr(t,n)  wgetn_wstr(stdscr,t,n)
#define hline_set(c,n)  whline_set(stdscr,c,n)
#define in_wch(c)       win_wch(stdscr,c)
#define in_wchnstr(c,n) win_wchnstr(stdscr,c,n)
#define in_wchstr(c)    win_wchstr(stdscr,c)
#define innwstr(c,n)    winnwstr(stdscr,c,n)
```

```
#define ins_nwstr(t,n)  wins_nwstr(stdscr,t,n)
#define ins_wch(c)      wins_wch(stdscr,c)
#define ins_wstr(t)     wins_wstr(stdscr,t)
#define inwstr(c)       winwstr(stdscr,c)
#define vline_set(c,n)  wvline_set(stdscr,c,n)
#define wadd_wchstr(win,str)    wadd_wchnstr(win,str,-1)
#define waddwstr(win,wstr)      waddnwstr(win,wstr,-1)
#define wget_wstr(w,t)  wgetn_wstr(w,t,-1)
#define win_wchstr(w,c) win_wchnstr(w,c,-1)
#define wins_wstr(w,t)  wins_nwstr(w,t,-1)
#define wgetbkgrnd(win,wch)     (*wch = win->_bkgrnd, OK)
#define mvadd_wch(y,x,c)        mvwadd_wch(stdscr,y,x,c)
#define mvadd_wchnstr(y,x,s,n)  mvwadd_wchnstr(stdscr,y,x,s,n)
#define mvadd_wchstr(y,x,s)     mvwadd_wchstr(stdscr,y,x,s)
#define mvaddnwstr(y,x,wstr,n)  mvwaddnwstr(stdscr,y,x,wstr,n)
#define mvaddwstr(y,x,wstr)     mvwaddwstr(stdscr,y,x,wstr)
#define mvget_wch(y,x,c)        mvwget_wch(stdscr,y,x,c)
#define mvget_wstr(y,x,t)       mvwget_wstr(stdscr,y,x,t)
#define mvgetn_wstr(y,x,t,n)    mvwgetn_wstr(stdscr,y,x,t,n)
#define mvhline_set(y,x,c,n)    mvwhline_set(stdscr,y,x,c,n)
#define mvin_wch(y,x,c) mvwin_wch(stdscr,y,x,c)
#define mvin_wchnstr(y,x,c,n)   mvwin_wchnstr(stdscr,y,x,c,n)
#define mvin_wchstr(y,x,c)      mvwin_wchstr(stdscr,y,x,c)
#define mvinnwstr(y,x,c,n)      mvwinnwstr(stdscr,y,x,c,n)
#define mvins_nwstr(y,x,t,n)    mvwins_nwstr(stdscr,y,x,t,n)
#define mvins_wch(y,x,c)        mvwins_wch(stdscr,y,x,c)
#define mvins_wstr(y,x,t)       mvwins_wstr(stdscr,y,x,t)
#define mvinwstr(y,x,c) mvwinwstr(stdscr,y,x,c)
#define mvvline_set(y,x,c,n)    mvwvline_set(stdscr,y,x,c,n)
#define  mvwadd_wch(win,y,x,c)     (wmove(win,y,x)  ==  ERR  ?  ERR  :
wadd_wch(win,c))
#define  mvwadd_wchnstr(win,y,x,s,n)       (wmove(win,y,x)  ==  ERR  ?
ERR  :  wadd_wchnstr(win,s,n))
#define  mvwadd_wchstr(win,y,x,s)          (wmove(win,y,x)  ==  ERR  ?
ERR  :  wadd_wchstr(win,s))
#define  mvwaddnwstr(win,y,x,wstr,n)       (wmove(win,y,x)  ==  ERR  ?
ERR  :  waddnwstr(win,wstr,n))
#define  mvwaddwstr(win,y,x,wstr)          (wmove(win,y,x)  ==  ERR  ?
ERR  :  waddwstr(win,wstr))
#define  mvwget_wch(win,y,x,c)     (wmove(win,y,x)  ==  ERR  ?  ERR  :
wget_wch(win,c))
#define  mvwget_wstr(win,y,x,t)    (wmove(win,y,x)  ==  ERR  ?  ERR  :
wget_wstr(win,t))
#define  mvwgetn_wstr(win,y,x,t,n)         (wmove(win,y,x)  ==  ERR  ?
ERR  :  wgetn_wstr(win,t,n))
#define  mvwhline_set(win,y,x,c,n)         (wmove(win,y,x)  ==  ERR  ?
ERR  :  whline_set(win,c,n))
#define  mvwin_wch(win,y,x,c)      (wmove(win,y,x)  ==  ERR  ?  ERR  :
win_wch(win,c))
#define  mvwin_wchnstr(win,y,x,c,n)        (wmove(win,y,x)  ==  ERR  ?
ERR  :  win_wchnstr(win,c,n))
#define  mvwin_wchstr(win,y,x,c)  (wmove(win,y,x)  ==  ERR  ?  ERR  :
win_wchstr(win,c))
#define  mvwinnwstr(win,y,x,c,n)  (wmove(win,y,x)  ==  ERR  ?  ERR  :
winnwstr(win,c,n))
#define  mvwins_nwstr(win,y,x,t,n)         (wmove(win,y,x)  ==  ERR  ?
ERR  :  wins_nwstr(win,t,n))
#define  mvwins_wch(win,y,x,c)     (wmove(win,y,x)  ==  ERR  ?  ERR  :
wins_wch(win,c))
#define  mvwins_wstr(win,y,x,t)    (wmove(win,y,x)  ==  ERR  ?  ERR  :
wins_wstr(win,t))
#define  mvwinwstr(win,y,x,c)      (wmove(win,y,x)  ==  ERR  ?  ERR  :
winwstr(win,c))
#define  mvwvline_set(win,y,x,c,n)         (wmove(win,y,x)  ==  ERR  ?
ERR  :  wvline_set(win,c,n))
#define NCURSES_MOUSE_MASK(b,m) ((m) << (((b) - 1) * 6))
```

```
#define NCURSES_BUTTON_RELEASED 001L
#define NCURSES_BUTTON_PRESSED  002L
#define NCURSES_BUTTON_CLICKED  004L
#define NCURSES_DOUBLE_CLICKED  010L
#define NCURSES_TRIPLE_CLICKED  020L
#define NCURSES_RESERVED_EVENT  040L
#define   BUTTON1_RELEASED                    NCURSES_MOUSE_MASK(1,
NCURSES_BUTTON_RELEASED)
#define             BUTTON1_PRESSED           NCURSES_MOUSE_MASK(1,
NCURSES_BUTTON_PRESSED)
#define             BUTTON1_CLICKED           NCURSES_MOUSE_MASK(1,
NCURSES_BUTTON_CLICKED)
#define      BUTTON1_DOUBLE_CLICKED           NCURSES_MOUSE_MASK(1,
NCURSES_DOUBLE_CLICKED)
#define      BUTTON1_TRIPLE_CLICKED           NCURSES_MOUSE_MASK(1,
NCURSES_TRIPLE_CLICKED)
#define   BUTTON2_RELEASED                    NCURSES_MOUSE_MASK(2,
NCURSES_BUTTON_RELEASED)
#define             BUTTON2_PRESSED           NCURSES_MOUSE_MASK(2,
NCURSES_BUTTON_PRESSED)
#define             BUTTON2_CLICKED           NCURSES_MOUSE_MASK(2,
NCURSES_BUTTON_CLICKED)
#define      BUTTON2_DOUBLE_CLICKED           NCURSES_MOUSE_MASK(2,
NCURSES_DOUBLE_CLICKED)
#define      BUTTON2_TRIPLE_CLICKED           NCURSES_MOUSE_MASK(2,
NCURSES_TRIPLE_CLICKED)
#define   BUTTON3_RELEASED                    NCURSES_MOUSE_MASK(3,
NCURSES_BUTTON_RELEASED)
#define             BUTTON3_PRESSED           NCURSES_MOUSE_MASK(3,
NCURSES_BUTTON_PRESSED)
#define             BUTTON3_CLICKED           NCURSES_MOUSE_MASK(3,
NCURSES_BUTTON_CLICKED)
#define      BUTTON3_DOUBLE_CLICKED           NCURSES_MOUSE_MASK(3,
NCURSES_DOUBLE_CLICKED)
#define      BUTTON3_TRIPLE_CLICKED           NCURSES_MOUSE_MASK(3,
NCURSES_TRIPLE_CLICKED)
#define   BUTTON4_RELEASED                    NCURSES_MOUSE_MASK(4,
NCURSES_BUTTON_RELEASED)
#define             BUTTON4_PRESSED           NCURSES_MOUSE_MASK(4,
NCURSES_BUTTON_PRESSED)
#define             BUTTON4_CLICKED           NCURSES_MOUSE_MASK(4,
NCURSES_BUTTON_CLICKED)
#define      BUTTON4_DOUBLE_CLICKED           NCURSES_MOUSE_MASK(4,
NCURSES_DOUBLE_CLICKED)
#define      BUTTON4_TRIPLE_CLICKED           NCURSES_MOUSE_MASK(4,
NCURSES_TRIPLE_CLICKED)
#define   BUTTON5_RELEASED                    NCURSES_MOUSE_MASK(5,
NCURSES_BUTTON_RELEASED)
#define             BUTTON5_PRESSED           NCURSES_MOUSE_MASK(5,
NCURSES_BUTTON_PRESSED)
#define             BUTTON5_CLICKED           NCURSES_MOUSE_MASK(5,
NCURSES_BUTTON_CLICKED)
#define      BUTTON5_DOUBLE_CLICKED           NCURSES_MOUSE_MASK(5,
NCURSES_DOUBLE_CLICKED)
#define      BUTTON5_TRIPLE_CLICKED           NCURSES_MOUSE_MASK(5,
NCURSES_TRIPLE_CLICKED)
#define BUTTON_CTRL     NCURSES_MOUSE_MASK(6, 0001L)
#define BUTTON_SHIFT    NCURSES_MOUSE_MASK(6, 0002L)
#define BUTTON_ALT      NCURSES_MOUSE_MASK(6, 0004L)
#define REPORT_MOUSE_POSITION   NCURSES_MOUSE_MASK(6, 0010L)
#define      BUTTON1_RESERVED_EVENT           NCURSES_MOUSE_MASK(1,
NCURSES_RESERVED_EVENT)
#define      BUTTON2_RESERVED_EVENT           NCURSES_MOUSE_MASK(2,
NCURSES_RESERVED_EVENT)
#define      BUTTON3_RESERVED_EVENT           NCURSES_MOUSE_MASK(3,
NCURSES_RESERVED_EVENT)
```

```
#define     BUTTON4_RESERVED_EVENT          NCURSES_MOUSE_MASK(4,
NCURSES_RESERVED_EVENT)
#define ALL_MOUSE_EVENTS       (REPORT_MOUSE_POSITION - 1)
#define BUTTON_RELEASE(e,         x) ((e) & NCURSES_MOUSE_MASK(x,
001))
#define BUTTON_PRESS(e, x) ((e) & NCURSES_MOUSE_MASK(x, 002))
#define BUTTON_CLICK(e, x) ((e) & NCURSES_MOUSE_MASK(x, 004))
#define BUTTON_DOUBLE_CLICK(e,   x) ((e) & NCURSES_MOUSE_MASK(x,
010))
#define BUTTON_TRIPLE_CLICK(e,   x) ((e) & NCURSES_MOUSE_MASK(x,
020))
#define  BUTTON_RESERVED_EVENT(e,                    x)   ((e)   &
NCURSES_MOUSE_MASK(x, 040))
#define                                mouse_trafo(y,x,to_screen)
wmouse_trafo(stdscr,y,x,to_screen)
#define _tracech_t     _tracecchar_t
#define _tracech_t2    _tracecchar_t2
#define TRACE_DISABLE   0x0000
#define TRACE_TIMES     0x0001
#define TRACE_TPUTS     0x0002
#define TRACE_UPDATE    0x0004
#define TRACE_MOVE      0x0008
#define TRACE_CHARPUT   0x0010
#define TRACE_ORDINARY  0x001F
#define TRACE_CALLS     0x0020
#define TRACE_VIRTPUT   0x0040
#define TRACE_IEVENT    0x0080
#define TRACE_BITS      0x0100
#define TRACE_ICALLS    0x0200
#define TRACE_CCALLS    0x0400
#define TRACE_DATABASE  0x0800
#define TRACE_ATTRS     0x1000
#define TRACE_SHIFT     13
#define TRACE_MAXIMUM   ((1 << TRACE_SHIFT) - 1)
#define OPTIMIZE_MVCUR  0x01
#define OPTIMIZE_HASHMAP     0x02
#define OPTIMIZE_SCROLL 0x04
#define OPTIMIZE_ALL    0xff

typedef unsigned long int chtype;
typedef chtype attr_t;

struct pdat {
    short _pad_y;
    short _pad_x;
    short _pad_top;
    short _pad_left;
    short _pad_bottom;
    short _pad_right;
};
typedef struct screen SCREEN;
typedef struct _win_st WINDOW;
typedef unsigned long int mmask_t;
typedef unsigned char bool;

typedef unsigned char NCURSES_BOOL;
typedef int (*NCURSES_OUTC) (int);
typedef int (*NCURSES_WINDOW_CB) (WINDOW *, void *);
typedef int (*NCURSES_SCREEN_CB) (SCREEN *, void *);
struct _win_st {
    short _cury;             /* current cursor position */
    short _curx;
    short _maxy;               /* maximums of x and y, NOT window
size */
    short _maxx;
```

```
    short _begy;                /* screen coords of upper-left-hand
corner */
    short _begx;
    short _flags;              /* window state flags */
    attr_t _attrs;               /* current attribute for non-space
character */
    chtype _bkgd;               /* 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;
    short _yoffset;             /* real begy is _begy + _yoffset */
    cchar_t _bkgrnd;            /* current background char/attribute
pair */
};
extern int COLORS;
extern int COLOR_PAIRS;
extern int COLS;
extern int LINES;
extern chtype acs_map[];
extern int add_wch(cchar_t *);
extern int add_wchnstr(cchar_t *, int);
extern int add_wchstr(cchar_t *);
extern int addch(const chtype);
extern int addchnstr(const chtype *, int);
extern int addchstr(const chtype *);
extern int addnstr(const char *, int);
extern int addnwstr(wchar_t *, int);
extern int addstr(const char *);
extern int addwstr(wchar_t *);
extern int assume_default_colors(int, int);
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 attroff(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 bkgrnd(cchar_t *);
```



```
extern void bkgrndset(cchar_t *);
extern int border(chtype, chtype, chtype, chtype, chtype, chtype,
chtype,
                  chtype);
extern int border_set(cchar_t *, cchar_t *, cchar_t *, cchar_t *,
                      cchar_t *, cchar_t *, cchar_t *, cchar_t *);
extern int box(WINDOW *, chtype, chtype);
extern int box_set(WINDOW *, cchar_t *, cchar_t *);
extern unsigned char can_change_color(void);
extern int cbreak(void);
extern int chgat(int, attr_t, short, const void *);
extern int clear(void);
extern int clearok(WINDOW *, unsigned char);
extern int clrtobot(void);
extern int clrtoeol(void);
extern int color_content(short, short *, short *, short *);
extern int color_set(short, void *);
extern int copywin(const WINDOW *, WINDOW *, int, int, int, int,
int, int,
                   int);
extern int curs_set(int);
extern WINDOW *curscr;
extern const char *curses_version(void);
extern int def_prog_mode(void);
extern int def_shell_mode(void);
extern int define_key(const char *, int);
extern int delay_output(int);
extern int delch(void);
extern int deleteln(void);
extern void delscreen(SCREEN *);
extern int delwin(WINDOW *);
extern WINDOW *derwin(WINDOW *, int, int, int, int);
extern int doupdate(void);
extern WINDOW *dupwin(WINDOW *);
extern int echo(void);
extern int echo_wchar(cchar_t *);
extern int echochar(const chtype);
extern int endwin(void);
extern int erase(void);
extern char erasechar(void);
extern int erasewchar(wchar_t *);
extern void filter(void);
extern int flash(void);
extern int flushinp(void);
extern int get_wch(wint_t *);
extern int get_wstr(wint_t *);
extern chtype getbkgd(WINDOW *);
extern int getbkgrnd(cchar_t *);
extern int getcchar(cchar_t *, wchar_t *, attr_t *, short *, void
*);
extern int getch(void);
extern int getmouse(MEVENT *);
extern int getn_wstr(wint_t *, int);
extern int getnstr(char *, int);
extern int getstr(char *);
extern WINDOW *getwin(FILE *);
extern int halfdelay(int);
extern unsigned char has_colors(void);
extern unsigned char has_ic(void);
extern unsigned char has_il(void);
extern int has_key(void);
extern bool has_mouse(void);
extern int hline(chtype, int);
extern int hline_set(cchar_t *, int);
extern void idcok(WINDOW *, unsigned char);
extern int idlok(WINDOW *, unsigned char);
```

```
extern void immedok(WINDOW *, unsigned char);
extern int in_wch(cchar_t *);
extern int in_wchnstr(cchar_t *, int);
extern int in_wchstr(cchar_t *);
extern chtype inch(void);
extern int inchnstr(chtype *, int);
extern int inchstr(chtype *);
extern int init_color(short, short, short, short);
extern int init_pair(short, short, short);
extern WINDOW *initscr(void);
extern int innstr(char *, int);
extern int innwstr(wchar_t *, int);
extern int ins_nwstr(wchar_t *, int);
extern int ins_wch(cchar_t *);
extern int ins_wstr(wchar_t *);
extern int insch(chtype);
extern int insdelln(int);
extern int insertln(void);
extern int insnstr(const char *, int);
extern int insstr(const char *);
extern int instr(char *);
extern int intrflush(WINDOW *, unsigned char);
extern int inwstr(wchar_t *);
extern unsigned char is_linetouched(WINDOW *, int);
extern unsigned char is_wintouched(WINDOW *);
extern unsigned char isendwin(void);
extern char *key_name(wchar_t);
extern char *keybound(int, int);
extern char *keyname(int);
extern int keyok(int, unsigned char);
extern int keypad(WINDOW *, unsigned char);
extern char killchar(void);
extern int killwchar(wchar_t *);
extern int leaveok(WINDOW *, unsigned char);
extern char *longname(void);
extern int mcprint(void);
extern int meta(WINDOW *, unsigned char);
extern bool mouse_trafo(int *, int *, bool);
extern int mouseinterval(int);
extern mmask_t mousemask(mmask_t, mmask_t *);
extern int move(int, int);
extern int mvadd_wch(int, int, cchar_t *);
extern int mvadd_wchnstr(int, int, cchar_t *, int);
extern int mvadd_wchstr(int, int, cchar_t *);
extern int mvaddch(const chtype, const chtype, 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 mvaddnwstr(int, int, wchar_t *, int);
extern int mvaddstr(int, int, const char *);
extern int mvaddwstr(int, int, wchar_t *);
extern int mvchgat(int, int, int, attr_t, short, const void *);
extern int mvcur(int, int, int, int);
extern int mvdelch(int, int);
extern int mvderwin(WINDOW *, int, int);
extern int mvget_wch(int, int, wint_t *);
extern int mvget_wstr(int, int, wint_t *);
extern int mvgetch(int, int);
extern int mvgetn_wstr(int, int, wint_t *, int);
extern int mvgetnstr(int, int, char *, int);
extern int mvgetstr(int, int, char *);
extern int mvhline(int, int, chtype, int);
extern int mvhline_set(int, int, cchar_t *, int);
extern int mvin_wch(int, int, cchar_t *);
extern int mvin_wchnstr(int, int, cchar_t *, int);
extern int mvin_wchstr(int, int, cchar_t *);
```



```
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 mvinnwstr(int, int, wchar_t *, int);
extern int mvins_nwstr(int, int, wchar_t *, int);
extern int mvins_wch(int, int, cchar_t *);
extern int mvins_wstr(int, int, wchar_t *);
extern int mvinsch(int, int, chtype);
extern int mvinsnstr(int, int, const char *, int);
extern int mvinsstr(int, int, const char *);
extern int mvinstr(int, int, char *);
extern int mvinwstr(int, int, wchar_t *);
extern int mvprintw(int, int, const char *, ...);
extern int mvscanw(int, int, char *, ...);
extern int mvvline(int, int, chtype, int);
extern int mvvline_set(int, int, cchar_t *, int);
extern int mvwadd_wch(WINDOW *, int, int, cchar_t *);
extern int mvwadd_wchnstr(WINDOW *, int, int, cchar_t *, int);
extern int mvwadd_wchstr(WINDOW *, int, int, cchar_t *);
extern int mvwaddch(const chtype, const chtype, const chtype,
                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 mvwaddnwstr(WINDOW *, int, int, wchar_t *, int);
extern int mvwaddstr(WINDOW *, int, int, const char *);
extern int mvwaddwstr(WINDOW *, int, int, wchar_t *);
extern int mvwchgat(WINDOW *, int, int, int, attr_t, short, const
void *);
extern int mvwdelch(WINDOW *, int, int);
extern int mvwget_wch(WINDOW *, int, int, wint_t *);
extern int mvwget_wstr(WINDOW *, int, int, wint_t *);
extern int mvwgetch(WINDOW *, int, int);
extern int mvwgetn_wstr(WINDOW *, int, int, wint_t *, int);
extern int mvwgetnstr(WINDOW *, int, int, char *, int);
extern int mvwgetstr(WINDOW *, int, int, char *);
extern int mvwhline(WINDOW *, int, int, chtype, int);
extern int mvwhline_set(WINDOW *, int, int, cchar_t *, int);
extern int mvwin(WINDOW *, int, int);
extern int mvwin_wch(WINDOW *, int, int, cchar_t *);
extern int mvwin_wchnstr(WINDOW *, int, int, cchar_t *, int);
extern int mvwin_wchstr(WINDOW *, int, int, cchar_t *);
extern chtype mvwinch(WINDOW *, int, int);
extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
extern int mvwinchstr(WINDOW *, int, int, chtype *);
extern int mvwinnstr(WINDOW *, int, int, char *, int);
extern int mvwinnwstr(WINDOW *, int, int, wchar_t *, int);
extern int mvwins_nwstr(WINDOW *, int, int, wchar_t *, int);
extern int mvwins_wch(WINDOW *, int, int, cchar_t *);
extern int mvwins_wstr(WINDOW *, int, int, wchar_t *);
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 mvwinstr(WINDOW *, int, int, char *);
extern int mvwinwstr(WINDOW *, int, int, wchar_t *);
extern int mvwprintw(WINDOW *, int, int, const char *, ...);
extern int mvwscanw(WINDOW *, int, int, char *, ...);
extern int mvwvline(WINDOW *, int, int, chtype, int);
extern int mvwvline_set(WINDOW *, int, int, cchar_t *, int);
extern int napms(int);
extern WINDOW *newpad(int, int);
extern WINDOW *newscr;
extern SCREEN *newterm(char *, FILE *, FILE *);
extern WINDOW *newwin(int, int, int, int);
extern int nl(void);
```

```
extern int nocbreak(void);
extern int nodelay(WINDOW *, unsigned char);
extern int noecho(void);
extern int nonl(void);
extern void noqiflush(void);
extern int noraw(void);
extern int notimeout(WINDOW *, unsigned char);
extern int overlay(const WINDOW *, WINDOW *);
extern int overwrite(const WINDOW *, WINDOW *);
extern int pair_content(short, short *, short *);
extern int pecho_wchar(WINDOW *, cchar_t *);
extern int pechochar(const chtype, const chtype);
extern int pnoutrefresh(WINDOW *, int, int, int, int, int, int);
extern int prefresh(WINDOW *, int, int, int, int, int, int);
extern int printw(const char *, ...);
extern int putwin(WINDOW *, FILE *);
extern void qiflush(void);
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 resetty(void);
extern int resizeterm(int, int);
extern int ripoffline(int, int (*)(WINDOW *, int));
extern int savetty(void);
extern int scanw(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 scrl(int);
extern int scroll(WINDOW *);
extern int scrollok(WINDOW *, unsigned char);
extern SCREEN *set_term(SCREEN *);
extern int setcchar(cchar_t *, wchar_t *, attr_t, short, void *);
extern int setscrreg(int, int);
extern attr_t slk_attr(void);
extern int slk_attr_off(const attr_t, void *);
extern int slk_attr_on(attr_t, void *);
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 char *slk_label(int);
extern int slk_noutrefresh(void);
extern int slk_refresh(void);
extern int slk_restore(void);
extern int slk_set(int, const char *, int);
extern int slk_touch(void);
extern int slk_wset(int, const wchar_t *, int);
extern int standend(void);
extern int standout(void);
extern int start_color(void);
extern WINDOW *stdscr;
extern WINDOW *subpad(WINDOW *, int, int, int, int);
extern WINDOW *subwin(WINDOW *, int, int, int, int);
extern int syncok(WINDOW *, unsigned char);
extern attr_t term_attrs(void);
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 char *unctrl(chtype);
extern int unget_wch(wchar_t);
extern int ungetch(int);
extern int ungetmouse(MEVENT *);
extern int untouchwin(WINDOW *);
extern int use_default_colors(void);
extern void use_env(unsigned char);
extern int use_extended_names(unsigned char);
extern int vid_attr(attr_t, short, void *);
extern int vid_puts(attr_t, short, void *, int);
extern int vidattr(chtype);
extern int vidputs(chtype, NCURSES_OUTC);
extern int vline(chtype, int);
extern int vline_set(cchar_t *, int);
extern int vw_printw(WINDOW *, const char *, va_list);
extern int vw_scanw(WINDOW *, char *, va_list);
extern int vwprintw(WINDOW *, const char *, va_list);
extern int vwscanw(WINDOW *, char *, va_list);
extern int wadd_wch(WINDOW *, cchar_t *);
extern int wadd_wchnstr(WINDOW *, cchar_t *, int);
extern int wadd_wchstr(WINDOW *, cchar_t *);
extern int waddch(WINDOW *, chtype);
extern int waddchnstr(WINDOW *, const chtype *, int);
extern int waddchstr(WINDOW *, const chtype *);
extern int waddnstr(WINDOW *, const char *, int);
extern int waddnwstr(WINDOW *, wchar_t *, int);
extern int waddstr(WINDOW *, const char *);
extern int waddwstr(WINDOW *, wchar_t *);
extern int wattr_get(WINDOW *, attr_t *, short *, void *);
extern int wattr_off(WINDOW *, attr_t, void *);
extern int wattr_on(WINDOW *, attr_t, void *);
extern int wattr_set(WINDOW *, attr_t, short, void *);
extern int wattroff(WINDOW *, int);
extern int wattron(WINDOW *, int);
extern int wattrset(WINDOW *, int);
extern int wbkgd(WINDOW *, chtype);
extern void wbkgdset(WINDOW *, chtype);
extern int wbkgrnd(WINDOW *, cchar_t *);
extern void wbkgrndset(WINDOW *, cchar_t *);
extern int wborder(WINDOW *, chtype, chtype, chtype, chtype, chtype,
                   chtype, chtype, chtype);
extern int wborder_set(WINDOW *, cchar_t *, cchar_t *, cchar_t *,
                       cchar_t *, cchar_t *, cchar_t *, cchar_t *,
                       cchar_t *);
extern int wchgat(WINDOW *, int, attr_t, short, const void *);
extern int wclear(WINDOW *);
extern int wclrtobot(WINDOW *);
extern int wclrtoeol(WINDOW *);
extern int wcolor_set(WINDOW *, short, void *);
extern void wcursyncup(WINDOW *);
extern int wdelch(WINDOW *);
extern int wdeleteln(WINDOW *);
extern int wecho_wchar(WINDOW *, cchar_t *);
extern int wechochar(const chtype, const chtype);
extern int werase(WINDOW *);
extern int wget_wch(WINDOW *, wint_t *);
extern int wget_wstr(WINDOW *, wint_t *);
extern int wgetbkgrnd(WINDOW *, cchar_t *);
extern int wgetch(WINDOW *);
extern int wgetn_wstr(WINDOW *, wint_t *, int);
extern int wgetnstr(WINDOW *, char *, int);
extern int wgetstr(WINDOW *, char *);
extern int whline(WINDOW *, chtype, int);
extern int whline_set(WINDOW *, cchar_t *, int);
```

```
extern int win_wch(WINDOW *, cchar_t *);
extern int win_wchnstr(WINDOW *, cchar_t *, int);
extern int win_wchstr(WINDOW *, cchar_t *);
extern chtype winch(WINDOW *);
extern int winchnstr(WINDOW *, chtype *, int);
extern int winchstr(WINDOW *, chtype *);
extern int winnstr(WINDOW *, char *, int);
extern int winnwstr(WINDOW *, wchar_t *, int);
extern int wins_nwstr(WINDOW *, wchar_t *, int);
extern int wins_wch(WINDOW *, cchar_t *);
extern int wins_wstr(WINDOW *, wchar_t *);
extern int winsch(WINDOW *, chtype);
extern int winsdelln(WINDOW *, int);
extern int winsertln(WINDOW *);
extern int winsnstr(WINDOW *, const char *, int);
extern int winsstr(WINDOW *, const char *);
extern int winstr(WINDOW *, char *);
extern int winwstr(WINDOW *, wchar_t *);
extern unsigned char wmouse_trafo(const WINDOW *, int *, int *,
bool);
extern int wmove(WINDOW *, int, int);
extern int wnoutrefresh(WINDOW *);
extern int wprintw(WINDOW *, const char *, ...);
extern int wredrawln(WINDOW *, int, int);
extern int wrefresh(WINDOW *);
extern int wresize(WINDOW *, int, int);
extern int wscanw(WINDOW *, char *, ...);
extern int wscrl(WINDOW *, int);
extern int wsetscrreg(WINDOW *, int, int);
extern int wstandend(WINDOW *);
extern int wstandout(WINDOW *);
extern void wsyncdown(WINDOW *);
extern void wsyncup(WINDOW *);
extern void wtimeout(WINDOW *, int);
extern int wtouchln(WINDOW *, int, int, int);
extern wchar_t *wunctrl(cchar_t *);
extern int wvline(WINDOW *, chtype, int);
extern int wvline_set(WINDOW *, cchar_t *, int);
```

15.9.2 ncursesw/ncurses_dll.h

```
#define NCURSES_API
#define NCURSES_IMPEXP
#define NCURSES_STATIC
#define NCURSES_WRAPPED_VAR(type,name)          extern     type
NCURSES_PUBLIC_VAR(name)(void)
#define NCURSES_PUBLIC_VAR(name)        _nc_ ##name
```

15.9.3 ncursesw/term.h

```
#define NCURSES_TERM_H_incl    1
#define NCURSES_VERSION "5.9"
#define NCURSES_SBOOL   char
#define NCURSES_XNAMES  1
#define TERMIOS 1
#define TTY     struct termios
#define TCSANOW TCSETA
#define TCSADRAIN       TCSETAW
#define TCSAFLUSH       TCSETAF
#define tcsetattr(fd, cmd, arg) ioctl(fd, cmd, arg)
#define tcgetattr(fd, arg)      ioctl(fd, TCGETA, arg)
#define cfgetospeed(t)  ((t)->c_cflag & CBAUD)
#define TCOFLUSH        1
```



```
#define TCIOFLUSH      2
#define tcflush(fd, arg)       ioctl(fd, TCFLSH, arg)
#define GET_TTY(fd, buf)       tcgetattr(fd, buf)
#define SET_TTY(fd, buf)       tcsetattr(fd, TCSADRAIN, buf)
#define NAMESIZE       256
#define CUR    cur_term->type.
#define auto_left_margin       CUR Booleans[0]
#define auto_right_margin      CUR Booleans[1]
#define no_esc_ctlc    CUR Booleans[2]
#define ceol_standout_glitch    CUR Booleans[3]
#define eat_newline_glitch     CUR Booleans[4]
#define erase_overstrike       CUR Booleans[5]
#define generic_type    CUR Booleans[6]
#define hard_copy      CUR Booleans[7]
#define has_meta_key    CUR Booleans[8]
#define has_status_line CUR Booleans[9]
#define insert_null_glitch     CUR Booleans[10]
#define memory_above    CUR Booleans[11]
#define memory_below    CUR Booleans[12]
#define move_insert_mode       CUR Booleans[13]
#define move_standout_mode     CUR Booleans[14]
#define over_strike    CUR Booleans[15]
#define status_line_esc_ok     CUR Booleans[16]
#define dest_tabs_magic_smso    CUR Booleans[17]
#define tilde_glitch    CUR Booleans[18]
#define transparent_underline   CUR Booleans[19]
#define xon_xoff       CUR Booleans[20]
#define needs_xon_xoff  CUR Booleans[21]
#define prtr_silent    CUR Booleans[22]
#define hard_cursor    CUR Booleans[23]
#define non_rev_rmcup   CUR Booleans[24]
#define no_pad_char    CUR Booleans[25]
#define non_dest_scroll_region  CUR Booleans[26]
#define can_change     CUR Booleans[27]
#define back_color_erase       CUR Booleans[28]
#define hue_lightness_saturation       CUR Booleans[29]
#define col_addr_glitch CUR Booleans[30]
#define cr_cancels_micro_mode   CUR Booleans[31]
#define has_print_wheel CUR Booleans[32]
#define row_addr_glitch CUR Booleans[33]
#define semi_auto_right_margin  CUR Booleans[34]
#define cpi_changes_res CUR Booleans[35]
#define lpi_changes_res CUR Booleans[36]
#define columns CUR Numbers[0]
#define init_tabs      CUR Numbers[1]
#define lines   CUR Numbers[2]
#define lines_of_memory CUR Numbers[3]
#define magic_cookie_glitch     CUR Numbers[4]
#define padding_baud_rate      CUR Numbers[5]
#define virtual_terminal       CUR Numbers[6]
#define width_status_line      CUR Numbers[7]
#define num_labels     CUR Numbers[8]
#define label_height    CUR Numbers[9]
#define label_width    CUR Numbers[10]
#define max_attributes  CUR Numbers[11]
#define maximum_windows CUR Numbers[12]
#define max_colors     CUR Numbers[13]
#define max_pairs      CUR Numbers[14]
#define no_color_video  CUR Numbers[15]
#define buffer_capacity CUR Numbers[16]
#define dot_vert_spacing       CUR Numbers[17]
#define dot_horz_spacing       CUR Numbers[18]
#define max_micro_address      CUR Numbers[19]
#define max_micro_jump  CUR Numbers[20]
#define micro_col_size  CUR Numbers[21]
#define micro_line_size CUR Numbers[22]
```

```
#define number_of_pins  CUR Numbers[23]
#define output_res_char CUR Numbers[24]
#define output_res_line CUR Numbers[25]
#define output_res_horz_inch    CUR Numbers[26]
#define output_res_vert_inch    CUR Numbers[27]
#define print_rate      CUR Numbers[28]
#define wide_char_size  CUR Numbers[29]
#define buttons CUR Numbers[30]
#define bit_image_entwining     CUR Numbers[31]
#define bit_image_type  CUR Numbers[32]
#define back_tab        CUR Strings[0]
#define bell    CUR Strings[1]
#define carriage_return CUR Strings[2]
#define change_scroll_region    CUR Strings[3]
#define clear_all_tabs  CUR Strings[4]
#define clear_screen    CUR Strings[5]
#define clr_eol CUR Strings[6]
#define clr_eos CUR Strings[7]
#define column_address  CUR Strings[8]
#define command_character       CUR Strings[9]
#define cursor_address  CUR Strings[10]
#define cursor_down     CUR Strings[11]
#define cursor_home     CUR Strings[12]
#define cursor_invisible        CUR Strings[13]
#define cursor_left     CUR Strings[14]
#define cursor_mem_address      CUR Strings[15]
#define cursor_normal   CUR Strings[16]
#define cursor_right    CUR Strings[17]
#define cursor_to_ll    CUR Strings[18]
#define cursor_up       CUR Strings[19]
#define cursor_visible  CUR Strings[20]
#define delete_character        CUR Strings[21]
#define delete_line     CUR Strings[22]
#define dis_status_line CUR Strings[23]
#define down_half_line  CUR Strings[24]
#define enter_alt_charset_mode  CUR Strings[25]
#define enter_blink_mode        CUR Strings[26]
#define enter_bold_mode CUR Strings[27]
#define enter_ca_mode   CUR Strings[28]
#define enter_delete_mode       CUR Strings[29]
#define enter_dim_mode  CUR Strings[30]
#define enter_insert_mode       CUR Strings[31]
#define enter_secure_mode       CUR Strings[32]
#define enter_protected_mode    CUR Strings[33]
#define enter_reverse_mode      CUR Strings[34]
#define enter_standout_mode     CUR Strings[35]
#define enter_underline_mode    CUR Strings[36]
#define erase_chars     CUR Strings[37]
#define exit_alt_charset_mode   CUR Strings[38]
#define exit_attribute_mode     CUR Strings[39]
#define exit_ca_mode    CUR Strings[40]
#define exit_delete_mode        CUR Strings[41]
#define exit_insert_mode        CUR Strings[42]
#define exit_standout_mode      CUR Strings[43]
#define exit_underline_mode     CUR Strings[44]
#define flash_screen    CUR Strings[45]
#define form_feed       CUR Strings[46]
#define from_status_line        CUR Strings[47]
#define init_1string    CUR Strings[48]
#define init_2string    CUR Strings[49]
#define init_3string    CUR Strings[50]
#define init_file       CUR Strings[51]
#define insert_character        CUR Strings[52]
#define insert_line     CUR Strings[53]
#define insert_padding  CUR Strings[54]
#define key_backspace   CUR Strings[55]
```



```
#define key_catab      CUR Strings[56]
#define key_clear      CUR Strings[57]
#define key_ctab       CUR Strings[58]
#define key_dc  CUR Strings[59]
#define key_dl  CUR Strings[60]
#define key_down       CUR Strings[61]
#define key_eic CUR Strings[62]
#define key_eol CUR Strings[63]
#define key_eos CUR Strings[64]
#define key_f0  CUR Strings[65]
#define key_f1  CUR Strings[66]
#define key_f10 CUR Strings[67]
#define key_f2  CUR Strings[68]
#define key_f3  CUR Strings[69]
#define key_f4  CUR Strings[70]
#define key_f5  CUR Strings[71]
#define key_f6  CUR Strings[72]
#define key_f7  CUR Strings[73]
#define key_f8  CUR Strings[74]
#define key_f9  CUR Strings[75]
#define key_home       CUR Strings[76]
#define key_ic  CUR Strings[77]
#define key_il  CUR Strings[78]
#define key_left       CUR Strings[79]
#define key_ll  CUR Strings[80]
#define key_npage      CUR Strings[81]
#define key_ppage      CUR Strings[82]
#define key_right      CUR Strings[83]
#define key_sf  CUR Strings[84]
#define key_sr  CUR Strings[85]
#define key_stab       CUR Strings[86]
#define key_up  CUR Strings[87]
#define keypad_local    CUR Strings[88]
#define keypad_xmit     CUR Strings[89]
#define lab_f0  CUR Strings[90]
#define lab_f1  CUR Strings[91]
#define lab_f10 CUR Strings[92]
#define lab_f2  CUR Strings[93]
#define lab_f3  CUR Strings[94]
#define lab_f4  CUR Strings[95]
#define lab_f5  CUR Strings[96]
#define lab_f6  CUR Strings[97]
#define lab_f7  CUR Strings[98]
#define lab_f8  CUR Strings[99]
#define lab_f9  CUR Strings[100]
#define meta_off       CUR Strings[101]
#define meta_on CUR Strings[102]
#define newline CUR Strings[103]
#define pad_char       CUR Strings[104]
#define parm_dch       CUR Strings[105]
#define parm_delete_line       CUR Strings[106]
#define parm_down_cursor       CUR Strings[107]
#define parm_ich       CUR Strings[108]
#define parm_index     CUR Strings[109]
#define parm_insert_line       CUR Strings[110]
#define parm_left_cursor       CUR Strings[111]
#define parm_right_cursor      CUR Strings[112]
#define parm_rindex    CUR Strings[113]
#define parm_up_cursor  CUR Strings[114]
#define pkey_key       CUR Strings[115]
#define pkey_local     CUR Strings[116]
#define pkey_xmit      CUR Strings[117]
#define print_screen    CUR Strings[118]
#define prtr_off       CUR Strings[119]
#define prtr_on CUR Strings[120]
#define repeat_char    CUR Strings[121]
```

```
#define reset_1string   CUR Strings[122]
#define reset_2string   CUR Strings[123]
#define reset_3string   CUR Strings[124]
#define reset_file      CUR Strings[125]
#define restore_cursor  CUR Strings[126]
#define row_address     CUR Strings[127]
#define save_cursor     CUR Strings[128]
#define scroll_forward  CUR Strings[129]
#define scroll_reverse  CUR Strings[130]
#define set_attributes  CUR Strings[131]
#define set_tab CUR Strings[132]
#define set_window      CUR Strings[133]
#define tab     CUR Strings[134]
#define to_status_line  CUR Strings[135]
#define underline_char  CUR Strings[136]
#define up_half_line    CUR Strings[137]
#define init_prog       CUR Strings[138]
#define key_a1  CUR Strings[139]
#define key_a3  CUR Strings[140]
#define key_b2  CUR Strings[141]
#define key_c1  CUR Strings[142]
#define key_c3  CUR Strings[143]
#define prtr_non        CUR Strings[144]
#define char_padding    CUR Strings[145]
#define acs_chars       CUR Strings[146]
#define plab_norm       CUR Strings[147]
#define key_btab        CUR Strings[148]
#define enter_xon_mode  CUR Strings[149]
#define exit_xon_mode   CUR Strings[150]
#define enter_am_mode   CUR Strings[151]
#define exit_am_mode    CUR Strings[152]
#define xon_character   CUR Strings[153]
#define xoff_character  CUR Strings[154]
#define ena_acs CUR Strings[155]
#define label_on        CUR Strings[156]
#define label_off       CUR Strings[157]
#define key_beg CUR Strings[158]
#define key_cancel      CUR Strings[159]
#define key_close       CUR Strings[160]
#define key_command     CUR Strings[161]
#define key_copy        CUR Strings[162]
#define key_create      CUR Strings[163]
#define key_end CUR Strings[164]
#define key_enter       CUR Strings[165]
#define key_exit        CUR Strings[166]
#define key_find        CUR Strings[167]
#define key_help        CUR Strings[168]
#define key_mark        CUR Strings[169]
#define key_message     CUR Strings[170]
#define key_move        CUR Strings[171]
#define key_next        CUR Strings[172]
#define key_open        CUR Strings[173]
#define key_options     CUR Strings[174]
#define key_previous    CUR Strings[175]
#define key_print       CUR Strings[176]
#define key_redo        CUR Strings[177]
#define key_reference   CUR Strings[178]
#define key_refresh     CUR Strings[179]
#define key_replace     CUR Strings[180]
#define key_restart     CUR Strings[181]
#define key_resume      CUR Strings[182]
#define key_save        CUR Strings[183]
#define key_suspend     CUR Strings[184]
#define key_undo        CUR Strings[185]
#define key_sbeg        CUR Strings[186]
#define key_scancel     CUR Strings[187]
```

```
#define key_scommand    CUR Strings[188]
#define key_scopy       CUR Strings[189]
#define key_screate     CUR Strings[190]
#define key_sdc CUR Strings[191]
#define key_sdl CUR Strings[192]
#define key_select      CUR Strings[193]
#define key_send        CUR Strings[194]
#define key_seol        CUR Strings[195]
#define key_sexit       CUR Strings[196]
#define key_sfind       CUR Strings[197]
#define key_shelp       CUR Strings[198]
#define key_shome       CUR Strings[199]
#define key_sic CUR Strings[200]
#define key_sleft       CUR Strings[201]
#define key_smessage    CUR Strings[202]
#define key_smove       CUR Strings[203]
#define key_snext       CUR Strings[204]
#define key_soptions    CUR Strings[205]
#define key_sprevious   CUR Strings[206]
#define key_sprint      CUR Strings[207]
#define key_sredo       CUR Strings[208]
#define key_sreplace    CUR Strings[209]
#define key_sright      CUR Strings[210]
#define key_srsume      CUR Strings[211]
#define key_ssave       CUR Strings[212]
#define key_ssuspend    CUR Strings[213]
#define key_sundo       CUR Strings[214]
#define req_for_input   CUR Strings[215]
#define key_f11 CUR Strings[216]
#define key_f12 CUR Strings[217]
#define key_f13 CUR Strings[218]
#define key_f14 CUR Strings[219]
#define key_f15 CUR Strings[220]
#define key_f16 CUR Strings[221]
#define key_f17 CUR Strings[222]
#define key_f18 CUR Strings[223]
#define key_f19 CUR Strings[224]
#define key_f20 CUR Strings[225]
#define key_f21 CUR Strings[226]
#define key_f22 CUR Strings[227]
#define key_f23 CUR Strings[228]
#define key_f24 CUR Strings[229]
#define key_f25 CUR Strings[230]
#define key_f26 CUR Strings[231]
#define key_f27 CUR Strings[232]
#define key_f28 CUR Strings[233]
#define key_f29 CUR Strings[234]
#define key_f30 CUR Strings[235]
#define key_f31 CUR Strings[236]
#define key_f32 CUR Strings[237]
#define key_f33 CUR Strings[238]
#define key_f34 CUR Strings[239]
#define key_f35 CUR Strings[240]
#define key_f36 CUR Strings[241]
#define key_f37 CUR Strings[242]
#define key_f38 CUR Strings[243]
#define key_f39 CUR Strings[244]
#define key_f40 CUR Strings[245]
#define key_f41 CUR Strings[246]
#define key_f42 CUR Strings[247]
#define key_f43 CUR Strings[248]
#define key_f44 CUR Strings[249]
#define key_f45 CUR Strings[250]
#define key_f46 CUR Strings[251]
#define key_f47 CUR Strings[252]
#define key_f48 CUR Strings[253]
```

```
#define key_f49 CUR Strings[254]
#define key_f50 CUR Strings[255]
#define key_f51 CUR Strings[256]
#define key_f52 CUR Strings[257]
#define key_f53 CUR Strings[258]
#define key_f54 CUR Strings[259]
#define key_f55 CUR Strings[260]
#define key_f56 CUR Strings[261]
#define key_f57 CUR Strings[262]
#define key_f58 CUR Strings[263]
#define key_f59 CUR Strings[264]
#define key_f60 CUR Strings[265]
#define key_f61 CUR Strings[266]
#define key_f62 CUR Strings[267]
#define key_f63 CUR Strings[268]
#define clr_bol CUR Strings[269]
#define clear_margins   CUR Strings[270]
#define set_left_margin CUR Strings[271]
#define set_right_margin       CUR Strings[272]
#define label_format    CUR Strings[273]
#define set_clock      CUR Strings[274]
#define display_clock   CUR Strings[275]
#define remove_clock    CUR Strings[276]
#define create_window   CUR Strings[277]
#define goto_window     CUR Strings[278]
#define hangup  CUR Strings[279]
#define dial_phone      CUR Strings[280]
#define quick_dial      CUR Strings[281]
#define tone    CUR Strings[282]
#define pulse   CUR Strings[283]
#define flash_hook      CUR Strings[284]
#define fixed_pause     CUR Strings[285]
#define wait_tone      CUR Strings[286]
#define user0   CUR Strings[287]
#define user1   CUR Strings[288]
#define user2   CUR Strings[289]
#define user3   CUR Strings[290]
#define user4   CUR Strings[291]
#define user5   CUR Strings[292]
#define user6   CUR Strings[293]
#define user7   CUR Strings[294]
#define user8   CUR Strings[295]
#define user9   CUR Strings[296]
#define orig_pair      CUR Strings[297]
#define orig_colors    CUR Strings[298]
#define initialize_color       CUR Strings[299]
#define initialize_pair CUR Strings[300]
#define set_color_pair  CUR Strings[301]
#define set_foreground  CUR Strings[302]
#define set_background  CUR Strings[303]
#define change_char_pitch      CUR Strings[304]
#define change_line_pitch      CUR Strings[305]
#define change_res_horz CUR Strings[306]
#define change_res_vert CUR Strings[307]
#define define_char    CUR Strings[308]
#define enter_doublewide_mode   CUR Strings[309]
#define enter_draft_quality    CUR Strings[310]
#define enter_italics_mode     CUR Strings[311]
#define enter_leftward_mode    CUR Strings[312]
#define enter_micro_mode       CUR Strings[313]
#define enter_near_letter_quality      CUR Strings[314]
#define enter_normal_quality    CUR Strings[315]
#define enter_shadow_mode      CUR Strings[316]
#define enter_subscript_mode    CUR Strings[317]
#define enter_superscript_mode  CUR Strings[318]
#define enter_upward_mode      CUR Strings[319]
```



```
#define exit_doublewide_mode    CUR Strings[320]
#define exit_italics_mode       CUR Strings[321]
#define exit_leftward_mode      CUR Strings[322]
#define exit_micro_mode CUR Strings[323]
#define exit_shadow_mode        CUR Strings[324]
#define exit_subscript_mode     CUR Strings[325]
#define exit_superscript_mode   CUR Strings[326]
#define exit_upward_mode        CUR Strings[327]
#define micro_column_address    CUR Strings[328]
#define micro_down      CUR Strings[329]
#define micro_left      CUR Strings[330]
#define micro_right     CUR Strings[331]
#define micro_row_address       CUR Strings[332]
#define micro_up        CUR Strings[333]
#define order_of_pins   CUR Strings[334]
#define parm_down_micro CUR Strings[335]
#define parm_left_micro CUR Strings[336]
#define parm_right_micro        CUR Strings[337]
#define parm_up_micro   CUR Strings[338]
#define select_char_set CUR Strings[339]
#define set_bottom_margin       CUR Strings[340]
#define set_bottom_margin_parm  CUR Strings[341]
#define set_left_margin_parm    CUR Strings[342]
#define set_right_margin_parm   CUR Strings[343]
#define set_top_margin  CUR Strings[344]
#define set_top_margin_parm     CUR Strings[345]
#define start_bit_image CUR Strings[346]
#define start_char_set_def      CUR Strings[347]
#define stop_bit_image  CUR Strings[348]
#define stop_char_set_def       CUR Strings[349]
#define subscript_characters    CUR Strings[350]
#define superscript_characters  CUR Strings[351]
#define these_cause_cr  CUR Strings[352]
#define zero_motion     CUR Strings[353]
#define char_set_names  CUR Strings[354]
#define key_mouse       CUR Strings[355]
#define mouse_info      CUR Strings[356]
#define req_mouse_pos   CUR Strings[357]
#define get_mouse       CUR Strings[358]
#define set_a_foreground        CUR Strings[359]
#define set_a_background        CUR Strings[360]
#define pkey_plab       CUR Strings[361]
#define device_type     CUR Strings[362]
#define code_set_init   CUR Strings[363]
#define set0_des_seq    CUR Strings[364]
#define set1_des_seq    CUR Strings[365]
#define set2_des_seq    CUR Strings[366]
#define set3_des_seq    CUR Strings[367]
#define set_lr_margin   CUR Strings[368]
#define set_tb_margin   CUR Strings[369]
#define bit_image_repeat        CUR Strings[370]
#define bit_image_newline       CUR Strings[371]
#define bit_image_carriage_return       CUR Strings[372]
#define color_names     CUR Strings[373]
#define define_bit_image_region CUR Strings[374]
#define end_bit_image_region    CUR Strings[375]
#define set_color_band  CUR Strings[376]
#define set_page_length CUR Strings[377]
#define display_pc_char CUR Strings[378]
#define enter_pc_charset_mode   CUR Strings[379]
#define exit_pc_charset_mode    CUR Strings[380]
#define enter_scancode_mode     CUR Strings[381]
#define exit_scancode_mode      CUR Strings[382]
#define pc_term_options CUR Strings[383]
#define scancode_escape CUR Strings[384]
#define alt_scancode_esc        CUR Strings[385]
```

```
#define enter_horizontal_hl_mode       CUR Strings[386]
#define enter_left_hl_mode     CUR Strings[387]
#define enter_low_hl_mode      CUR Strings[388]
#define enter_right_hl_mode    CUR Strings[389]
#define enter_top_hl_mode      CUR Strings[390]
#define enter_vertical_hl_mode  CUR Strings[391]
#define set_a_attributes       CUR Strings[392]
#define set_pglen_inch  CUR Strings[393]
#define BOOLWRITE      37
#define NUMWRITE       33
#define STRWRITE       394
#define beehive_glitch  no_esc_ctlc
#define teleray_glitch  dest_tabs_magic_smso
#define micro_char_size micro_col_size
#define termcap_init2   CUR Strings[394]
#define termcap_reset   CUR Strings[395]
#define magic_cookie_glitch_ul  CUR Numbers[33]
#define backspaces_with_bs     CUR Booleans[37]
#define crt_no_scrolling       CUR Booleans[38]
#define no_correctly_working_cr CUR Booleans[39]
#define carriage_return_delay   CUR Numbers[34]
#define new_line_delay  CUR Numbers[35]
#define linefeed_if_not_lf     CUR Strings[396]
#define backspace_if_not_bs    CUR Strings[397]
#define gnu_has_meta_key       CUR Booleans[40]
#define linefeed_is_newline    CUR Booleans[41]
#define backspace_delay CUR Numbers[36]
#define horizontal_tab_delay    CUR Numbers[37]
#define number_of_function_keys CUR Numbers[38]
#define other_non_function_keys CUR Strings[398]
#define arrow_key_map   CUR Strings[399]
#define has_hardware_tabs      CUR Booleans[42]
#define return_does_clr_eol    CUR Booleans[43]
#define acs_ulcorner    CUR Strings[400]
#define acs_llcorner    CUR Strings[401]
#define acs_urcorner    CUR Strings[402]
#define acs_lrcorner    CUR Strings[403]
#define acs_ltee       CUR Strings[404]
#define acs_rtee       CUR Strings[405]
#define acs_btee       CUR Strings[406]
#define acs_ttee       CUR Strings[407]
#define acs_hline      CUR Strings[408]
#define acs_vline      CUR Strings[409]
#define acs_plus       CUR Strings[410]
#define memory_lock    CUR Strings[411]
#define memory_unlock   CUR Strings[412]
#define box_chars_1    CUR Strings[413]
#define BOOLCOUNT      44
#define NUMCOUNT       39
#define STRCOUNT       414
#define acs_chars_index 146
#define cur_term       NCURSES_PUBLIC_VAR(cur_term())
#define boolnames      NCURSES_PUBLIC_VAR(boolnames())
#define boolcodes      NCURSES_PUBLIC_VAR(boolcodes())
#define boolfnames     NCURSES_PUBLIC_VAR(boolfnames())
#define numnames       NCURSES_PUBLIC_VAR(numnames())
#define numcodes       NCURSES_PUBLIC_VAR(numcodes())
#define numfnames      NCURSES_PUBLIC_VAR(numfnames())
#define strnames       NCURSES_PUBLIC_VAR(strnames())
#define strcodes       NCURSES_PUBLIC_VAR(strcodes())
#define strfnames      NCURSES_PUBLIC_VAR(strfnames())

typedef struct termtype {
    char *term_names;
    char *str_table;
    char *Booleans;
```



```
    short *Numbers;
    char **Strings;
    char *ext_str_table;
    char **ext_Names;
    unsigned short num_Booleans;
    unsigned short num_Numbers;
    unsigned short num_Strings;
    unsigned short ext_Booleans;
    unsigned short ext_Numbers;
    unsigned short ext_Strings;
} TERMTYPE;
typedef struct term {
    TERMTYPE type;
    short Filedes;
    struct termios Ottyb;
    struct termios Nttyb;
    int _baudrate;
    char *_termname;
} TERMINAL;
extern TERMINAL *cur_term;
extern int del_curterm(TERMINAL *);
extern int putp(const char *);
extern int restartterm(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 *);
extern char *tgetstr(char *, char **);
extern char *tgoto(const char *, int, int);
extern int tigetflag(char *);
extern int tigetnum(char *);
extern char *tigetstr(char *);
extern char *tparm(char *, ...);
extern int tputs(const char *, int, int (*)(int));
extern char ttytype[];
```

15.9.4 ncursesw/unctrl.h

```
#define NCURSES_UNCTRL_H_incl   1
#define NCURSES_VERSION "5.9"
```

15.10 Interface Definitions for libncursesw

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

Other interfaces listed in Section 15.8 shall behave as described in the referenced base document.

15.11 Interfaces for libutil

Table 15-11 defines the library name and shared object name for the libutil library

Table 15-11 libutil Definition

Library:	libutil
SONAME:	libutil.so.1

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

[LSB] This Specification

15.11.1 Utility Functions

15.11.1.1 Interfaces for Utility Functions

An LSB conforming implementation shall provide the generic functions for Utility Functions specified in Table 15-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 15-12 libutil - Utility Functions Function Interfaces

forkpty [LSB]	login [LSB]	login_tty [LSB]	logout [LSB]
logwtmp [LSB]	openpty [LSB]		

15.12 Data Definitions for libutil

This section defines global identifiers and their values that are associated with interfaces contained in libutil. 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.

15.12.1 pty.h

```
extern int forkpty(int *__amaster, char *__name,
                   const struct termios *__termp,
                   const struct winsize *__winp);
extern int openpty(int *__amaster, int *__aslave, char *__name,
                   const struct termios *__termp,
                   const struct winsize *__winp);
```

15.13 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 15.11 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, const struct termios * termp,
const struct winsize * winp);
```

Description

The `forkpty()` function shall find and open a pseudo-terminal device pair in the same manner as the `openpty()` function. If a pseudo-terminal is available, `forkpty()` shall create a new process in the same manner as the `fork()` function, and prepares the new process for login in the same manner as `login_tty()`.

If *`termp`* is not null, it shall refer to a `termios` structure that shall be used to initialize the characteristics of the slave device. If *`winp`* is not null, it shall refer to a `winsize` structure used to initialize the window size of the slave device.

Return Value

On success, the parent process shall return the process id of the child, and the child shall return 0. On error, no new process shall be created, -1 shall be returned, and `errno` shall be set appropriately. On success, the parent process shall receive the file descriptor of the master side of the pseudo-terminal in the location referenced by *`amaster`*, and, if *`name`* is not NULL, the filename of the slave device in *`name`*.

Errors

EAGAIN

Unable to create a new process.

ENOENT

There are no available pseudo-terminals.

ENOMEM

Insufficient memory was available.

login

Name

`login` — login utility function

Synopsis

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

Description

The `login()` function shall update the user accounting databases. The *ut* parameter shall reference a `utmp` structure for all fields except the following:

1. The *ut_type* field shall be set to `USER_PROCESS`.
2. The *ut_pid* field shall be set to the process identifier for the current process.
3. The *ut_line* field shall be set to the name of the controlling terminal device. The name shall be found by examining the device associated with the standard input, output and error streams in sequence, until one associated with a terminal device is found. If none of these streams refers to a terminal device, the *ut_line* field shall be set to `"???"`. If the terminal device is in the `/dev` directory hierarchy, the *ut_line* field shall not contain the leading `"/dev/"`, otherwise it shall be set to the final component of the pathname of the device. If the user accounting database imposes a limit on the size of the *ut_line* field, it shall truncate the name, but any such limit shall not be smaller than `UT_LINESIZE` (including a terminating null character).

Return Value

None

Errors

None

login_tty

Name

`login_tty` — Prepare a terminal for login

Synopsis

```
#include <utmp.h>
int login_tty (int fdr);
```

Description

The `login_tty()` function shall prepare the terminal device referenced by the file descriptor *`fdr`*. This function shall create a new session, make the terminal the controlling terminal for the current process, and set the standard input, output, and error streams of the current process to the terminal. If *`fdr`* is not the standard input, output or error stream, then `login_tty()` shall close *`fdr`*.

Return Value

On success, `login_tty()` shall return zero; otherwise -1 is returned, and errno shall be set appropriately.

Errors

ENOTTY

`fdr` does not refer to a terminal device.

logout

Name

`logout` — logout utility function

Synopsis

```
#include <utmp.h>
int logout (const char * line );
```

Description

Given the device *`line`*, the `logout()` function shall search the user accounting database which is read by `getutent()` for an entry with the corresponding line, and with the type of `USER_PROCESS`. If a corresponding entry is located, it shall be updated as follows:

1. The `ut_name` field shall be set to zeroes (`UT_NAMESIZE` NUL bytes).
2. The `ut_host` field shall be set to zeroes (`UT_HOSTSIZE` NUL bytes).
3. The `ut_tv` shall be set to the current time of day.
4. The `ut_type` field shall be set to `DEAD_PROCESS`.

Return Value

On success, the `logout()` function shall return non-zero. Zero is returned if there was no entry to remove, or if the utmp file could not be opened or updated.

logwtmp

Name

`logwtmp` — append an entry to the wtmp file

Synopsis

```
#include <utmp.h>
void logwtmp (const char * line , const char * name , const char *
host );
```

Description

If the process has permission to update the user accounting databases, the `logwtmp()` function shall append a record to the user accounting database that records all logins and logouts. The record to be appended shall be constructed as follows:

1. The `ut_line` field shall be initialized from *`line`*. If the user accounting database imposes a limit on the size of the *`ut_line`* field, it shall truncate the value, but any such limit shall not be smaller than `UT_LINESIZE` (including a terminating null character).
2. The `ut_name` field shall be initialized from *`name`*. If the user accounting database imposes a limit on the size of the *`ut_name`* field, it shall truncate the value, but any such limit shall not be smaller than `UT_NAMESIZE` (including a terminating null character).
3. The `ut_host` field shall be initialized from *`host`*. If the user accounting database imposes a limit on the size of the *`ut_host`* field, it shall truncate the value, but any such limit shall not be smaller than `UT_HOSTSIZE` (including a terminating null character).
4. If the *`name`* parameter does not refer to an empty string (i.e. `""`), the `ut_type` field shall be set to `USER_PROCESS`; otherwise the `ut_type` field shall be set to `DEAD_PROCESS`.
5. The `ut_id` field shall be set to the process identifier for the current process.
6. The `ut_tv` field shall be set to the current time of day.

Note: If a process does not have write access to the the user accounting database, the `logwtmp()` function will not update it. Since the function does not return any value, an application has no way of knowing whether it succeeded or failed.

Return Value

None.

openpty

Name

`openpty` — find and open an available pseudo-terminal

Synopsis

```
#include <pty.h>
int openpty(int *amaster, int *aslave, char *name, const struct termios
*termp, const struct winsize *winp);
```

Description

The `openpty()` function shall find an available pseudo-terminal and return file descriptors for the master and slave devices in the locations referenced by *amaster* and *aslave* respectively. If *name* is not NULL, the filename of the slave shall be placed in the user supplied buffer referenced by *name*. If *termp* is not NULL, it shall point to a `termios` structure used to initialize the terminal parameters of the slave pseudo-terminal device. If *winp* is not NULL, it shall point to a `winsize` structure used to initialize the window size parameters of the slave pseudo-terminal device.

Return Value

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

Errors

ENOENT

There are no available pseudo-terminals.

V C++ Libraries

16 Libraries

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

16.1 Interfaces for libstdcxx

Table 16-1 defines the library name and shared object name for the libstdcxx library

Table 16-1 libstdcxx Definition

Library:	libstdcxx
SONAME:	libstdc++.so.6

Unless stated otherwise, all symbols are in the `std::` namespace.

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

[CXXABI-1.86] Itanium™ C++ ABI
[ISOCXX] ISO/IEC 14882: 2003 C++ Language
[LSB] This Specification

16.1.1 C++ Runtime Support

16.1.1.1 Interfaces for C++ Runtime Support

An LSB conforming implementation shall provide the generic methods for C++ Runtime Support specified in Table 16-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-2 libstdcxx - C++ Runtime Support Function Interfaces

__gnu_cxx::__atomic_add(int volatile*, int)(GLIBCXX_3.4) [CXXABI-1.86]
__gnu_cxx::__exchange_and_add(int volatile*, int)(GLIBCXX_3.4) [CXXABI-1.86]
__gnu_cxx::__verbose_terminate_handler()(CXXABI_1.3) [CXXABI-1.86]
unexpected()(GLIBCXX_3.4) [ISOCXX]
set_terminate(void (*)())(GLIBCXX_3.4) [ISOCXX]
set_unexpected(void (*)())(GLIBCXX_3.4) [ISOCXX]
set_new_handler(void (*)())(GLIBCXX_3.4) [ISOCXX]
__throw_bad_cast()(GLIBCXX_3.4) [ISOCXX]
__throw_bad_alloc()(GLIBCXX_3.4) [ISOCXX]
__throw_bad_typeid()(GLIBCXX_3.4) [ISOCXX]
uncaught_exception()(GLIBCXX_3.4) [ISOCXX]
__throw_ios_failure(char const*)(GLIBCXX_3.4) [ISOCXX]

__throw_logic_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_range_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_domain_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_length_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_out_of_range(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_bad_exception()(GLIBCXX_3.4) [ISOCXX]
__throw_runtime_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_overflow_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_underflow_error(char const*)(GLIBCXX_3.4) [ISOCXX]
__throw_invalid_argument(char const*)(GLIBCXX_3.4) [ISOCXX]
terminate()(GLIBCXX_3.4) [ISOCXX]
operator delete[](void*)(GLIBCXX_3.4) [ISOCXX]
operator delete[](void*, nothrow_t const&)(GLIBCXX_3.4) [ISOCXX]
operator delete(void*)(GLIBCXX_3.4) [ISOCXX]
operator delete(void*, nothrow_t const&)(GLIBCXX_3.4) [ISOCXX]
__cxa_allocate_exception(CXXABI_1.3) [CXXABI-1.86]
__cxa_bad_cast(CXXABI_1.3) [CXXABI-1.86]
__cxa_bad_typeid(CXXABI_1.3) [CXXABI-1.86]
__cxa_begin_catch(CXXABI_1.3) [CXXABI-1.86]
__cxa_call_unexpected(CXXABI_1.3) [CXXABI-1.86]
__cxa_current_exception_type(CXXABI_1.3) [CXXABI-1.86]
__cxa_demangle(CXXABI_1.3) [CXXABI-1.86]
__cxa_end_catch(CXXABI_1.3) [CXXABI-1.86]
__cxa_free_exception(CXXABI_1.3) [CXXABI-1.86]
__cxa_get_exception_ptr(CXXABI_1.3.1) [CXXABI-1.86]
__cxa_get_globals(CXXABI_1.3) [CXXABI-1.86]
__cxa_get_globals_fast(CXXABI_1.3) [CXXABI-1.86]
__cxa_guard_abort(CXXABI_1.3) [CXXABI-1.86]
__cxa_guard_acquire(CXXABI_1.3) [CXXABI-1.86]
__cxa_guard_release(CXXABI_1.3) [CXXABI-1.86]
__cxa_pure_virtual(CXXABI_1.3) [CXXABI-1.86]
__cxa_rethrow(CXXABI_1.3) [CXXABI-1.86]
__cxa_throw(CXXABI_1.3) [CXXABI-1.86]

__cxa_vec_cctor(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_cleanup(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_ctor(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_delete(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_delete2(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_delete3(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_dtor(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_new(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_new2(CXXABI_1.3) [CXXABI-1.86]
__cxa_vec_new3(CXXABI_1.3) [CXXABI-1.86]
__dynamic_cast(CXXABI_1.3) [CXXABI-1.86]
__gxx_personality_v0(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for C++ Runtime Support specified in Table 16-3, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-3 libstdcxx - C++ Runtime Support Data Interfaces

cin(GLIBCXX_3.4) [ISOCXX]
cerr(GLIBCXX_3.4) [ISOCXX]
clog(GLIBCXX_3.4) [ISOCXX]
cout(GLIBCXX_3.4) [ISOCXX]
wcin(GLIBCXX_3.4) [ISOCXX]
wcerr(GLIBCXX_3.4) [ISOCXX]
wclog(GLIBCXX_3.4) [ISOCXX]
wcout(GLIBCXX_3.4) [ISOCXX]
nothrow(GLIBCXX_3.4) [ISOCXX]

16.1.2 C++ type descriptors for built-in types

16.1.2.1 Interfaces for C++ type descriptors for built-in types

No external methods are defined for libstdcxx - C++ type descriptors for built-in types in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for C++ type descriptors for built-in types specified in Table 16-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-4 libstdcxx - C++ type descriptors for built-in types Data Interfaces

typeinfo for signed char const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for bool const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for char const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for double const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long double const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for float const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned char const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for int const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned int const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for short const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned short const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for void const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for wchar_t const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned long long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for signed char*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for bool*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for char*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for double*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long double*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for float*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned char*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for int*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned int*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for short*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned short*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for void*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for wchar_t*(CXXABI_1.3) [CXXABI-1.86]

typeinfo for long long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned long long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo for signed char(CXXABI_1.3) [CXXABI-1.86]
typeinfo for bool(CXXABI_1.3) [CXXABI-1.86]
typeinfo for char(CXXABI_1.3) [CXXABI-1.86]
typeinfo for double(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long double(CXXABI_1.3) [CXXABI-1.86]
typeinfo for float(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned char(CXXABI_1.3) [CXXABI-1.86]
typeinfo for int(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned int(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned long(CXXABI_1.3) [CXXABI-1.86]
typeinfo for short(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned short(CXXABI_1.3) [CXXABI-1.86]
typeinfo for void(CXXABI_1.3) [CXXABI-1.86]
typeinfo for wchar_t(CXXABI_1.3) [CXXABI-1.86]
typeinfo for long long(CXXABI_1.3) [CXXABI-1.86]
typeinfo for unsigned long long(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for signed char const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for bool const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for char const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for double const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long double const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for float const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned char const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for int const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned int const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for short const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned short const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for void const*(CXXABI_1.3) [CXXABI-1.86]

typeinfo name for wchar_t const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned long long const*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for signed char*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for bool*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for char*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for double*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long double*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for float*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned char*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for int*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned int*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for short*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned short*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for void*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for wchar_t*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned long long*(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for signed char(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for bool(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for char(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for double(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long double(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for float(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned char(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for int(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned int(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned long(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for short(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned short(CXXABI_1.3) [CXXABI-1.86]

typeinfo name for void(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for wchar_t(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for long long(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for unsigned long long(CXXABI_1.3) [CXXABI-1.86]

16.1.3 C++ _Rb_tree

16.1.3.1 Interfaces for C++ _Rb_tree

An LSB conforming implementation shall provide the generic methods for C++ _Rb_tree specified in Table 16-5, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-5 libstdcxx - C++ _Rb_tree Function Interfaces

_Rb_tree_decrement(_Rb_tree_node_base const*)(GLIBCXX_3.4) [LSB]
_Rb_tree_decrement(_Rb_tree_node_base*)(GLIBCXX_3.4) [LSB]
_Rb_tree_increment(_Rb_tree_node_base const*)(GLIBCXX_3.4) [LSB]
_Rb_tree_increment(_Rb_tree_node_base*)(GLIBCXX_3.4) [LSB]
_Rb_tree_black_count(_Rb_tree_node_base const*, _Rb_tree_node_base const*)(GLIBCXX_3.4) [LSB]
_Rb_tree_rotate_left(_Rb_tree_node_base*, _Rb_tree_node_base*&)(GLIBCXX_3.4) [LSB]
_Rb_tree_rotate_right(_Rb_tree_node_base*, _Rb_tree_node_base*&)(GLIBCXX_3.4) [LSB]
_Rb_tree_rebalance_for_erase(_Rb_tree_node_base*, _Rb_tree_node_base&)(GLIBCXX_3.4) [LSB]
_Rb_tree_insert_and_rebalance(bool, _Rb_tree_node_base*, _Rb_tree_node_base*, _Rb_tree_node_base&)(GLIBCXX_3.4) [LSB]

16.1.4 Class type_info

16.1.4.1 Class data for type_info

The virtual table for the std::type_info class is described by Table 16-6

Table 16-6 Primary vtable for type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for type_info
vfunc[0]:	type_info::~type_info()
vfunc[1]:	type_info::~type_info()

vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const

The Run Time Type Information for the std::type_info class is described by Table 16-7

Table 16-7 typeinfo for type_info

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for type_info

16.1.4.2 Interfaces for Class type_info

An LSB conforming implementation shall provide the generic methods for Class std::type_info specified in Table 16-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-8 libstdcxx - Class type_info Function Interfaces

type_info::__do_catch(type_info const*, void**, unsigned int) const(GLIBCXX_3.4) [ISOCXX]
type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const(GLIBCXX_3.4) [ISOCXX]
type_info::__is_pointer_p() const(GLIBCXX_3.4) [ISOCXX]
type_info::__is_function_p() const(GLIBCXX_3.4) [ISOCXX]
type_info::~type_info()(GLIBCXX_3.4) [ISOCXX]
type_info::~type_info()(GLIBCXX_3.4) [ISOCXX]
type_info::~type_info()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::type_info specified in Table 16-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-9 libstdcxx - Class type_info Data Interfaces

typeinfo for type_info(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for type_info(GLIBCXX_3.4) [CXXABI-1.86]
vtable for type_info(GLIBCXX_3.4) [CXXABI-1.86]

16.1.5 Class __cxxabiv1::__enum_type_info

16.1.5.1 Class data for __cxxabiv1::__enum_type_info

The virtual table for the __cxxabiv1::__enum_type_info class is described by Table 16-10

Table 16-10 Primary vtable for __cxxabiv1::__enum_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__enum_type_info
vfunc[0]:	__cxxabiv1::__enum_type_info::~__enum_type_info()
vfunc[1]:	__cxxabiv1::__enum_type_info::~__enum_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const

The Run Time Type Information for the __cxxabiv1::__enum_type_info class is described by Table 16-11

Table 16-11 typeinfo for __cxxabiv1::__enum_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__enum_type_info

16.1.5.2 Interfaces for Class __cxxabiv1::__enum_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__enum_type_info specified in Table 16-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-12 libstdcxx - Class __cxxabiv1::__enum_type_info Function Interfaces

__cxxabiv1::__enum_type_info::~__enum_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__enum_type_info::~__enum_type_info()(CXXABI_1.3) [CXXABI-1.86]

__cxxabiv1::__enum_type_info::~__enum_type_info()(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__enum_type_info specified in Table 16-13, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-13 libstdcxx - Class __cxxabiv1::__enum_type_info Data Interfaces

typeinfo for __cxxabiv1::__enum_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__enum_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__enum_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.6 Class __cxxabiv1::__array_type_info

16.1.6.1 Class data for __cxxabiv1::__array_type_info

The virtual table for the __cxxabiv1::__array_type_info class is described by Table 16-14

Table 16-14 Primary vtable for __cxxabiv1::__array_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__array_type_info
vfunc[0]:	__cxxabiv1::__array_type_info::~__array_type_info()
vfunc[1]:	__cxxabiv1::__array_type_info::~__array_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const

The Run Time Type Information for the __cxxabiv1::__array_type_info class is described by Table 16-15

Table 16-15 typeinfo for __cxxabiv1::__array_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info

Name	typeinfo name for __cxxabiv1::__array_type_info

16.1.6.2 Interfaces for Class __cxxabiv1::__array_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__array_type_info specified in Table 16-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-16 libstdcxx - Class __cxxabiv1::__array_type_info Function Interfaces

__cxxabiv1::__array_type_info::~__array_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__array_type_info::~__array_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__array_type_info::~__array_type_info()(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__array_type_info specified in Table 16-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-17 libstdcxx - Class __cxxabiv1::__array_type_info Data Interfaces

typeinfo for __cxxabiv1::__array_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__array_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__array_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.7 Class __cxxabiv1::__class_type_info

16.1.7.1 Class data for __cxxabiv1::__class_type_info

The virtual table for the __cxxabiv1::__class_type_info class is described by Table 16-18

Table 16-18 Primary vtable for __cxxabiv1::__class_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__class_type_info
vfunc[0]:	__cxxabiv1::__class_type_info::~__class_type_info()
vfunc[1]:	__cxxabiv1::__class_type_info::~__class_type_info()
vfunc[2]:	type_info::__is_pointer_p() const

vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__class_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&) const

The Run Time Type Information for the __cxxabiv1::__class_type_info class is described by Table 16-19

Table 16-19 typeinfo for __cxxabiv1::__class_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__class_type_info

16.1.7.2 Interfaces for Class __cxxabiv1::__class_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__class_type_info specified in Table 16-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-20 libstdcxx - Class __cxxabiv1::__class_type_info Function Interfaces

__cxxabiv1::__class_type_info::~__class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__class_type_info::~__class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__class_type_info::~__class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__class_type_info::__do_catch(type_info const*, void**, unsigned int) const(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&) const(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__class_type_info specified in Table 16-21, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-21 libstdcxx - Class __cxxabiv1::__class_type_info Data Interfaces

typeinfo for __cxxabiv1::__class_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__class_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__class_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.8 Class __cxxabiv1::__pbase_type_info

16.1.8.1 Class data for __cxxabiv1::__pbase_type_info

The virtual table for the __cxxabiv1::__pbase_type_info class is described by Table 16-22

Table 16-22 Primary vtable for __cxxabiv1::__pbase_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__pbase_type_info
vfunc[0]:	__cxxabiv1::__pbase_type_info::~__pbase_type_info()
vfunc[1]:	__cxxabiv1::__pbase_type_info::~__pbase_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__pbase_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const

The Run Time Type Information for the __cxxabiv1::__pbase_type_info class is described by Table 16-23

Table 16-23 typeinfo for __cxxabiv1::__pbase_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info

Name	typeinfo name for __cxxabiv1::__pbase_type_info

16.1.8.2 Interfaces for Class __cxxabiv1::__pbase_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__pbase_type_info specified in Table 16-24, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-24 libstdcxx - Class __cxxabiv1::__pbase_type_info Function Interfaces

__cxxabiv1::__pbase_type_info::~__pbase_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pbase_type_info::~__pbase_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pbase_type_info::~__pbase_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pbase_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__pbase_type_info specified in Table 16-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-25 libstdcxx - Class __cxxabiv1::__pbase_type_info Data Interfaces

typeinfo for __cxxabiv1::__pbase_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__pbase_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__pbase_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.9 Class __cxxabiv1::__pointer_type_info

16.1.9.1 Class data for __cxxabiv1::__pointer_type_info

The virtual table for the __cxxabiv1::__pointer_type_info class is described by Table 16-26

Table 16-26 Primary vtable for __cxxabiv1::__pointer_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__pointer_type_info

vfunc[0]:	__cxxabiv1::__pointer_type_info::~__pointer_type_info()
vfunc[1]:	__cxxabiv1::__pointer_type_info::~__pointer_type_info()
vfunc[2]:	__cxxabiv1::__pointer_type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__pointer_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const

The Run Time Type Information for the __cxxabiv1::__pointer_type_info class is described by Table 16-27

Table 16-27 typeinfo for __cxxabiv1::__pointer_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__pointer_type_info

16.1.9.2 Interfaces for Class __cxxabiv1::__pointer_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__pointer_type_info specified in Table 16-28, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-28 libstdcxx - Class __cxxabiv1::__pointer_type_info Function Interfaces

__cxxabiv1::__pointer_type_info::~__pointer_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pointer_type_info::~__pointer_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pointer_type_info::~__pointer_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pointer_type_info::__is_pointer_p() const(CXXABI_1.3) [CXXABI-1.86]

__cxxabiv1::__pointer_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__pointer_type_info specified in Table 16-29, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-29 libstdcxx - Class __cxxabiv1::__pointer_type_info Data Interfaces

typeinfo for __cxxabiv1::__pointer_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__pointer_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__pointer_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.10 Class __cxxabiv1::__function_type_info

16.1.10.1 Class data for __cxxabiv1::__function_type_info

The virtual table for the __cxxabiv1::__function_type_info class is described by Table 16-30

Table 16-30 Primary vtable for __cxxabiv1::__function_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__function_type_info
vfunc[0]:	__cxxabiv1::__function_type_info::~__function_type_info()
vfunc[1]:	__cxxabiv1::__function_type_info::~__function_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	__cxxabiv1::__function_type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const

The Run Time Type Information for the __cxxabiv1::__function_type_info class is described by Table 16-31

Table 16-31 typeinfo for __cxxabiv1::__function_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info

Name	typeinfo name for __cxxabiv1::__function_type_info

16.1.10.2 Interfaces for Class __cxxabiv1::__function_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__function_type_info specified in Table 16-32, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-32 libstdcxx - Class __cxxabiv1::__function_type_info Function Interfaces

__cxxabiv1::__function_type_info::~__function_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__function_type_info::~__function_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__function_type_info::~__function_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__function_type_info::__is_function_p() const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__function_type_info specified in Table 16-33, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-33 libstdcxx - Class __cxxabiv1::__function_type_info Data Interfaces

typeinfo for __cxxabiv1::__function_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__function_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__function_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.11 Class __cxxabiv1::__si_class_type_info

16.1.11.1 Class data for __cxxabiv1::__si_class_type_info

The virtual table for the __cxxabiv1::__si_class_type_info class is described by Table 16-34

Table 16-34 Primary vtable for __cxxabiv1::__si_class_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__si_class_type_info
vfunc[0]:	__cxxabiv1::__si_class_type_info::~__si_class_type_info()

vfunc[1]:	__cxxabiv1::__si_class_type_info::~__si_class_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__class_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__si_class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&) const

The Run Time Type Information for the __cxxabiv1::__si_class_type_info class is described by Table 16-35

Table 16-35 typeinfo for __cxxabiv1::__si_class_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__si_class_type_info

16.1.11.2 Interfaces for Class __cxxabiv1::__si_class_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__si_class_type_info specified in Table 16-36, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-36 libstdcxx - Class __cxxabiv1::__si_class_type_info Function Interfaces

__cxxabiv1::__si_class_type_info::~__si_class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__si_class_type_info::~__si_class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__si_class_type_info::~__si_class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__si_class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&) const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__si_class_type_info specified in Table 16-37, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-37 libstdcxx - Class __cxxabiv1::__si_class_type_info Data Interfaces

typeinfo for __cxxabiv1::__si_class_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__si_class_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__si_class_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.12 Class __cxxabiv1::__vmi_class_type_info

16.1.12.1 Class data for __cxxabiv1::__vmi_class_type_info

The virtual table for the __cxxabiv1::__vmi_class_type_info class is described by Table 16-38

Table 16-38 Primary vtable for __cxxabiv1::__vmi_class_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__vmi_class_type_info
vfunc[0]:	__cxxabiv1::__vmi_class_type_info::~__vmi_class_type_info()
vfunc[1]:	__cxxabiv1::__vmi_class_type_info::~__vmi_class_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__class_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	__cxxabiv1::__class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__vmi_class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&) const

The Run Time Type Information for the __cxxabiv1::__vmi_class_type_info class is described by Table 16-39

Table 16-39 typeinfo for __cxxabiv1::__vmi_class_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__vmi_class_type_info

16.1.12.2 Interfaces for Class __cxxabiv1::__vmi_class_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__vmi_class_type_info specified in Table 16-40, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-40 libstdcxx - Class __cxxabiv1::__vmi_class_type_info Function Interfaces

__cxxabiv1::__vmi_class_type_info::~__vmi_class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__vmi_class_type_info::~__vmi_class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__vmi_class_type_info::~__vmi_class_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__vmi_class_type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__upcast_result&) const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__vmi_class_type_info specified in Table 16-41, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-41 libstdcxx - Class __cxxabiv1::__vmi_class_type_info Data Interfaces

typeinfo for __cxxabiv1::__vmi_class_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__vmi_class_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__vmi_class_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.13 Class __cxxabiv1::__fundamental_type_info

16.1.13.1 Class data for __cxxabiv1::__fundamental_type_info

The virtual table for the __cxxabiv1::__fundamental_type_info class is described by Table 16-42

Table 16-42 Primary vtable for __cxxabiv1::__fundamental_type_info

Base Offset	0

Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__fundamental_type_info
vfunc[0]:	__cxxabiv1::__fundamental_type_info::~__fundamental_type_info()
vfunc[1]:	__cxxabiv1::__fundamental_type_info::~__fundamental_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const

The Run Time Type Information for the __cxxabiv1::__fundamental_type_info class is described by Table 16-43

Table 16-43 typeinfo for __cxxabiv1::__fundamental_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__fundamental_type_info

16.1.13.2 Interfaces for Class __cxxabiv1::__fundamental_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__fundamental_type_info specified in Table 16-44, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-44 libstdcxx - Class __cxxabiv1::__fundamental_type_info Function Interfaces

__cxxabiv1::__fundamental_type_info::~__fundamental_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__fundamental_type_info::~__fundamental_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__fundamental_type_info::~__fundamental_type_info()(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__fundamental_type_info specified in Table 16-45, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-45 libstdcxx - Class __cxxabiv1::__fundamental_type_info Data Interfaces

typeinfo for __cxxabiv1::__fundamental_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__fundamental_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__fundamental_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.14 Class __cxxabiv1::__pointer_to_member_type_info

16.1.14.1 Class data for __cxxabiv1::__pointer_to_member_type_info

The virtual table for the __cxxabiv1::__pointer_to_member_type_info class is described by Table 16-46

Table 16-46 Primary vtable for __cxxabiv1::__pointer_to_member_type_info

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __cxxabiv1::__pointer_to_member_type_info
vfunc[0]:	__cxxabiv1::__pointer_to_member_type_info::~__pointer_to_member_type_info()
vfunc[1]:	__cxxabiv1::__pointer_to_member_type_info::~__pointer_to_member_type_info()
vfunc[2]:	type_info::__is_pointer_p() const
vfunc[3]:	type_info::__is_function_p() const
vfunc[4]:	__cxxabiv1::__pbase_type_info::__do_catch(type_info const*, void**, unsigned int) const
vfunc[5]:	type_info::__do_upcast(__cxxabiv1::__class_type_info const*, void**) const
vfunc[6]:	__cxxabiv1::__pointer_to_member_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const

The Run Time Type Information for the __cxxabiv1::__pointer_to_member_type_info class is described by Table 16-47

Table 16-47 typeinfo for __cxxabiv1::__pointer_to_member_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__pointer_to_member_type_info

16.1.14.2 Interfaces for Class __cxxabiv1::__pointer_to_member_type_info

An LSB conforming implementation shall provide the generic methods for Class __cxxabiv1::__pointer_to_member_type_info specified in Table 16-48, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-48 libstdcxx - Class __cxxabiv1::__pointer_to_member_type_info Function Interfaces

__cxxabiv1::__pointer_to_member_type_info::~__pointer_to_member_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pointer_to_member_type_info::~__pointer_to_member_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pointer_to_member_type_info::~__pointer_to_member_type_info()(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__pointer_to_member_type_info::__pointer_catch(__cxxabiv1::__pbase_type_info const*, void**, unsigned int) const(CXXABI_1.3) [CXXABI-1.86]

An LSB conforming implementation shall provide the generic data interfaces for Class __cxxabiv1::__pointer_to_member_type_info specified in Table 16-49, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-49 libstdcxx - Class __cxxabiv1::__pointer_to_member_type_info Data Interfaces

typeinfo for __cxxabiv1::__pointer_to_member_type_info(CXXABI_1.3) [CXXABI-1.86]
typeinfo name for __cxxabiv1::__pointer_to_member_type_info(CXXABI_1.3) [CXXABI-1.86]
vtable for __cxxabiv1::__pointer_to_member_type_info(CXXABI_1.3) [CXXABI-1.86]

16.1.15 Class __gnu_cxx::stdio_filebuf<char, char_traits<char> >

16.1.15.1 Interfaces for Class __gnu_cxx::stdio_filebuf<char, char_traits<char> >

No external methods are defined for libstdcxx - Class __gnu_cxx::stdio_filebuf<char, std::char_traits<char> > in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class __gnu_cxx::stdio_filebuf<char, std::char_traits<char> > specified in Table 16-50, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-50 libstdcxx - Class __gnu_cxx::stdio_filebuf<char, char_traits<char> > Data Interfaces

typeinfo for __gnu_cxx::stdio_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __gnu_cxx::stdio_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.16 Class __gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t> >

16.1.16.1 Interfaces for Class __gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t> >

No external methods are defined for libstdcxx - Class __gnu_cxx::stdio_filebuf<wchar_t, std::char_traits<wchar_t> > in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class __gnu_cxx::stdio_filebuf<wchar_t, std::char_traits<wchar_t> > specified in Table 16-51, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-51 libstdcxx - Class __gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for __gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.17 Class __gnu_cxx::__pool_alloc_base

16.1.17.1 Interfaces for Class __gnu_cxx::__pool_alloc_base

An LSB conforming implementation shall provide the generic methods for Class __gnu_cxx::__pool_alloc_base specified in Table 16-52, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-52 libstdcxx - Class __gnu_cxx::__pool_alloc_base Function Interfaces

__gnu_cxx::__pool_alloc_base::_M_get_mutex()(GLIBCXX_3.4.2) [LSB]

16.1.18 Class __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >

16.1.18.1 Class data for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >

The virtual table for the __gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char> > class is described by Table 16-53

Table 16-53 Primary vtable for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >
vfunc[0]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::~stdio_sync_filebuf()
vfunc[1]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::~stdio_sync_filebuf()
vfunc[2]:	basic_streambuf<char, char_traits<char> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::sync()
vfunc[7]:	basic_streambuf<char, char_traits<char> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::underflow()

vfunc[10]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::uflow()
vfunc[11]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::pbackfail(int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::overflow(int)

16.1.18.2 Interfaces for Class __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class __gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char> > specified in Table 16-54, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-54 libstdcxx - Class __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> > Function Interfaces

__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::file()(GLIBCXX_3.4.2) [LSB]

An LSB conforming implementation shall provide the generic data interfaces for Class __gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char> > specified in Table 16-55, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-55 libstdcxx - Class __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> > Data Interfaces

typeinfo for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.19 Class __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >

16.1.19.1 Class data for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >

The virtual table for the __gnu_cxx::stdio_sync_filebuf<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-56

Table 16-56 Primary vtable for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >
vfunc[0]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::~stdio_sync_filebuf()
vfunc[1]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::~stdio_sync_filebuf()
vfunc[2]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::underflow()
vfunc[10]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)

16.1.19.2 Interfaces for Class __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class __gnu_cxx::stdio_sync_filebuf<wchar_t, std::char_traits<wchar_t> > specified in Table 16-57, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-57 libstdcxx - Class __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> > Function Interfaces

__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >::file()(GLIBCXX_3.4.2) [LSB]

An LSB conforming implementation shall provide the generic data interfaces for Class __gnu_cxx::stdio_sync_filebuf<wchar_t, std::char_traits<wchar_t> > specified in Table 16-58, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-58 libstdcxx - Class __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.20 Class exception

16.1.20.1 Class data for exception

The virtual table for the std::exception class is described by Table 16-59

Table 16-59 Primary vtable for exception

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for exception
vfunc[0]:	exception::~exception()
vfunc[1]:	exception::~exception()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::exception class is described by Table 16-60

Table 16-60 typeinfo for exception

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for exception

16.1.20.2 Interfaces for Class exception

An LSB conforming implementation shall provide the generic methods for Class std::exception specified in Table 16-61, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-61 libstdcxx - Class exception Function Interfaces

exception::what() const(GLIBCXX_3.4) [ISOCXX]
exception::~exception()(GLIBCXX_3.4) [ISOCXX]
exception::~exception()(GLIBCXX_3.4) [ISOCXX]
exception::~exception()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::exception specified in Table 16-62, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-62 libstdcxx - Class exception Data Interfaces

typeinfo for exception(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for exception(GLIBCXX_3.4) [CXXABI-1.86]
vtable for exception(GLIBCXX_3.4) [CXXABI-1.86]

16.1.21 Class bad_typeid

16.1.21.1 Class data for bad_typeid

The virtual table for the std::bad_typeid class is described by Table 16-63

Table 16-63 Primary vtable for bad_typeid

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for bad_typeid
vfunc[0]:	bad_typeid::~bad_typeid()
vfunc[1]:	bad_typeid::~bad_typeid()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad_typeid class is described by Table 16-64

Table 16-64 typeinfo for bad_typeid

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for bad_typeid

16.1.21.2 Interfaces for Class bad_typeid

An LSB conforming implementation shall provide the generic methods for Class std::bad_typeid specified in Table 16-65, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-65 libstdcxx - Class bad_typeid Function Interfaces

bad_typeid::~bad_typeid()(GLIBCXX_3.4) [ISOCXX]
bad_typeid::~bad_typeid()(GLIBCXX_3.4) [ISOCXX]
bad_typeid::~bad_typeid()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::bad_typeid specified in Table 16-66, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-66 libstdcxx - Class bad_typeid Data Interfaces

typeinfo for bad_typeid(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for bad_typeid(GLIBCXX_3.4) [CXXABI-1.86]
vtable for bad_typeid(GLIBCXX_3.4) [CXXABI-1.86]

16.1.22 Class logic_error

16.1.22.1 Class data for logic_error

The virtual table for the std::logic_error class is described by Table 16-67

Table 16-67 Primary vtable for logic_error

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for logic_error
vfunc[0]:	logic_error::~logic_error()
vfunc[1]:	logic_error::~logic_error()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::logic_error class is described by Table 16-68

Table 16-68 typeinfo for logic_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for logic_error

16.1.22.2 Interfaces for Class logic_error

An LSB conforming implementation shall provide the generic methods for Class std::logic_error specified in Table 16-69, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-69 libstdcxx - Class logic_error Function Interfaces

logic_error::what() const(GLIBCXX_3.4) [ISOCXX]
logic_error::logic_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
logic_error::logic_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
logic_error::~logic_error()(GLIBCXX_3.4) [ISOCXX]
logic_error::~logic_error()(GLIBCXX_3.4) [ISOCXX]
logic_error::~logic_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::logic_error specified in Table 16-70, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-70 libstdcxx - Class logic_error Data Interfaces

typeinfo for logic_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for logic_error(GLIBCXX_3.4) [CXXABI-1.86]
vtable for logic_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.23 Class range_error

16.1.23.1 Class data for range_error

The virtual table for the std::range_error class is described by Table 16-71

Table 16-71 Primary vtable for range_error

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for range_error
vfunc[0]:	range_error::~range_error()
vfunc[1]:	range_error::~range_error()

vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the std::range_error class is described by Table 16-72

Table 16-72 typeinfo for range_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for range_error

16.1.23.2 Interfaces for Class range_error

An LSB conforming implementation shall provide the generic methods for Class std::range_error specified in Table 16-73, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-73 libstdcxx - Class range_error Function Interfaces

range_error::range_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
range_error::range_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
range_error::~range_error()(GLIBCXX_3.4) [ISOCXX]
range_error::~range_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::range_error specified in Table 16-74, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-74 libstdcxx - Class range_error Data Interfaces

typeinfo for range_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for range_error(GLIBCXX_3.4) [CXXABI-1.86]
vtable for range_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.24 Class domain_error

16.1.24.1 Class data for domain_error

The virtual table for the std::domain_error class is described by Table 16-75

Table 16-75 Primary vtable for domain_error

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for domain_error
vfunc[0]:	domain_error::~domain_error()

vfunc[1]:	domain_error::~domain_error()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::domain_error class is described by Table 16-76

Table 16-76 typeinfo for domain_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for domain_error

16.1.24.2 Interfaces for Class domain_error

An LSB conforming implementation shall provide the generic methods for Class std::domain_error specified in Table 16-77, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-77 libstdcxx - Class domain_error Function Interfaces

domain_error::domain_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
domain_error::domain_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
domain_error::~domain_error()(GLIBCXX_3.4) [ISOCXX]
domain_error::~domain_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::domain_error specified in Table 16-78, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-78 libstdcxx - Class domain_error Data Interfaces

typeinfo for domain_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for domain_error(GLIBCXX_3.4) [CXXABI-1.86]
vtable for domain_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.25 Class length_error

16.1.25.1 Class data for length_error

The virtual table for the std::length_error class is described by Table 16-79

Table 16-79 Primary vtable for length_error

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for length_error

vfunc[0]:	length_error::~length_error()
vfunc[1]:	length_error::~length_error()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::length_error class is described by Table 16-80

Table 16-80 typeinfo for length_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for length_error

16.1.25.2 Interfaces for Class length_error

An LSB conforming implementation shall provide the generic methods for Class std::length_error specified in Table 16-81, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-81 libstdcxx - Class length_error Function Interfaces

length_error::length_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
length_error::length_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
length_error::~length_error()(GLIBCXX_3.4) [ISOCXX]
length_error::~length_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::length_error specified in Table 16-82, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-82 libstdcxx - Class length_error Data Interfaces

typeinfo for length_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for length_error(GLIBCXX_3.4) [CXXABI-1.86]
vtable for length_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.26 Class out_of_range

16.1.26.1 Class data for out_of_range

The virtual table for the std::out_of_range class is described by Table 16-83

Table 16-83 Primary vtable for out_of_range

Base Offset	0
Virtual Base Offset	0

RTTI	typeinfo for out_of_range
vfunc[0]:	out_of_range::~out_of_range()
vfunc[1]:	out_of_range::~out_of_range()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::out_of_range class is described by Table 16-84

Table 16-84 typeinfo for out_of_range

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for out_of_range

16.1.26.2 Interfaces for Class out_of_range

An LSB conforming implementation shall provide the generic methods for Class std::out_of_range specified in Table 16-85, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-85 libstdcxx - Class out_of_range Function Interfaces

out_of_range::out_of_range(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
out_of_range::out_of_range(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
out_of_range::~out_of_range()(GLIBCXX_3.4) [ISOCXX]
out_of_range::~out_of_range()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::out_of_range specified in Table 16-86, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-86 libstdcxx - Class out_of_range Data Interfaces

typeinfo for out_of_range(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for out_of_range(GLIBCXX_3.4) [CXXABI-1.86]
vtable for out_of_range(GLIBCXX_3.4) [CXXABI-1.86]

16.1.27 Class bad_exception

16.1.27.1 Class data for bad_exception

The virtual table for the std::bad_exception class is described by Table 16-87

Table 16-87 Primary vtable for bad_exception

Base Offset	0

Virtual Base Offset	0
RTTI	typeinfo for bad_exception
vfunc[0]:	bad_exception::~bad_exception()
vfunc[1]:	bad_exception::~bad_exception()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad_exception class is described by Table 16-88

Table 16-88 typeinfo for bad_exception

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for bad_exception

16.1.27.2 Interfaces for Class bad_exception

An LSB conforming implementation shall provide the generic methods for Class std::bad_exception specified in Table 16-89, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-89 libstdcxx - Class bad_exception Function Interfaces

bad_exception::~bad_exception()(GLIBCXX_3.4) [ISOCXX]
bad_exception::~bad_exception()(GLIBCXX_3.4) [ISOCXX]
bad_exception::~bad_exception()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::bad_exception specified in Table 16-90, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-90 libstdcxx - Class bad_exception Data Interfaces

typeinfo for bad_exception(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for bad_exception(GLIBCXX_3.4) [CXXABI-1.86]
vtable for bad_exception(GLIBCXX_3.4) [CXXABI-1.86]

16.1.28 Class runtime_error

16.1.28.1 Class data for runtime_error

The virtual table for the std::runtime_error class is described by Table 16-91

Table 16-91 Primary vtable for runtime_error

Base Offset	0
Virtual Base Offset	0

RTTI	typeinfo for runtime_error
vfunc[0]:	runtime_error::~runtime_error()
vfunc[1]:	runtime_error::~runtime_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the std::runtime_error class is described by Table 16-92

Table 16-92 typeinfo for runtime_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for runtime_error

16.1.28.2 Interfaces for Class runtime_error

An LSB conforming implementation shall provide the generic methods for Class std::runtime_error specified in Table 16-93, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-93 libstdcxx - Class runtime_error Function Interfaces

runtime_error::what() const(GLIBCXX_3.4) [ISOCXX]
runtime_error::runtime_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
runtime_error::runtime_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
runtime_error::~runtime_error()(GLIBCXX_3.4) [ISOCXX]
runtime_error::~runtime_error()(GLIBCXX_3.4) [ISOCXX]
runtime_error::~runtime_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::runtime_error specified in Table 16-94, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-94 libstdcxx - Class runtime_error Data Interfaces

typeinfo for runtime_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for runtime_error(GLIBCXX_3.4) [CXXABI-1.86]
vtable for runtime_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.29 Class overflow_error

16.1.29.1 Class data for overflow_error

The virtual table for the std::overflow_error class is described by Table 16-95

Table 16-95 Primary vtable for overflow_error

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for overflow_error
vfunc[0]:	overflow_error::~overflow_error()
vfunc[1]:	overflow_error::~overflow_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the std::overflow_error class is described by Table 16-96

Table 16-96 typeinfo for overflow_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for overflow_error

16.1.29.2 Interfaces for Class overflow_error

An LSB conforming implementation shall provide the generic methods for Class std::overflow_error specified in Table 16-97, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-97 libstdcxx - Class overflow_error Function Interfaces

overflow_error::overflow_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
overflow_error::overflow_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
overflow_error::~overflow_error()(GLIBCXX_3.4) [ISOCXX]
overflow_error::~overflow_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::overflow_error specified in Table 16-98, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-98 libstdcxx - Class overflow_error Data Interfaces

typeinfo for overflow_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for overflow_error(GLIBCXX_3.4) [CXXABI-1.86]
vtable for overflow_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.30 Class underflow_error

16.1.30.1 Class data for underflow_error

The virtual table for the std::underflow_error class is described by Table 16-99

Table 16-99 Primary vtable for underflow_error

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for underflow_error
vfunc[0]:	underflow_error::~underflow_error()
vfunc[1]:	underflow_error::~underflow_error()
vfunc[2]:	runtime_error::what() const

The Run Time Type Information for the std::underflow_error class is described by Table 16-100

Table 16-100 typeinfo for underflow_error

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for underflow_error

16.1.30.2 Interfaces for Class underflow_error

An LSB conforming implementation shall provide the generic methods for Class std::underflow_error specified in Table 16-101, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-101 libstdcxx - Class underflow_error Function Interfaces

underflow_error::underflow_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
underflow_error::underflow_error(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
underflow_error::~underflow_error()(GLIBCXX_3.4) [ISOCXX]
underflow_error::~underflow_error()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::underflow_error specified in Table 16-102, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-102 libstdcxx - Class underflow_error Data Interfaces

typeinfo for underflow_error(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for underflow_error(GLIBCXX_3.4) [CXXABI-1.86]

vtable for underflow_error(GLIBCXX_3.4) [CXXABI-1.86]

16.1.31 Class invalid_argument

16.1.31.1 Class data for invalid_argument

The virtual table for the std::invalid_argument class is described by Table 16-103

Table 16-103 Primary vtable for invalid_argument

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for invalid_argument
vfunc[0]:	invalid_argument::~invalid_argument()
vfunc[1]:	invalid_argument::~invalid_argument()
vfunc[2]:	logic_error::what() const

The Run Time Type Information for the std::invalid_argument class is described by Table 16-104

Table 16-104 typeinfo for invalid_argument

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for invalid_argument

16.1.31.2 Interfaces for Class invalid_argument

An LSB conforming implementation shall provide the generic methods for Class std::invalid_argument specified in Table 16-105, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-105 libstdcxx - Class invalid_argument Function Interfaces

invalid_argument::invalid_argument(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
invalid_argument::invalid_argument(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
invalid_argument::~invalid_argument()(GLIBCXX_3.4) [ISOCXX]
invalid_argument::~invalid_argument()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::invalid_argument specified in Table 16-106, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-106 libstdcxx - Class invalid_argument Data Interfaces

typeinfo for invalid_argument(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for invalid_argument(GLIBCXX_3.4) [CXXABI-1.86]
vtable for invalid_argument(GLIBCXX_3.4) [CXXABI-1.86]

16.1.32 Class bad_cast

16.1.32.1 Class data for bad_cast

The virtual table for the std::bad_cast class is described by Table 16-107

Table 16-107 Primary vtable for bad_cast

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for bad_cast
vfunc[0]:	bad_cast::~bad_cast()
vfunc[1]:	bad_cast::~bad_cast()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad_cast class is described by Table 16-108

Table 16-108 typeinfo for bad_cast

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for bad_cast

16.1.32.2 Interfaces for Class bad_cast

An LSB conforming implementation shall provide the generic methods for Class std::bad_cast specified in Table 16-109, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-109 libstdcxx - Class bad_cast Function Interfaces

bad_cast::~bad_cast()(GLIBCXX_3.4) [ISOCXX]
bad_cast::~bad_cast()(GLIBCXX_3.4) [ISOCXX]
bad_cast::~bad_cast()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::bad_cast specified in Table 16-110, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-110 libstdcxx - Class bad_cast Data Interfaces

typeinfo for bad_cast(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for bad_cast(GLIBCXX_3.4) [CXXABI-1.86]
vtable for bad_cast(GLIBCXX_3.4) [CXXABI-1.86]

16.1.33 Class bad_alloc

16.1.33.1 Class data for bad_alloc

The virtual table for the std::bad_alloc class is described by Table 16-111

Table 16-111 Primary vtable for bad_alloc

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for bad_alloc
vfunc[0]:	bad_alloc::~bad_alloc()
vfunc[1]:	bad_alloc::~bad_alloc()
vfunc[2]:	exception::what() const

The Run Time Type Information for the std::bad_alloc class is described by Table 16-112

Table 16-112 typeinfo for bad_alloc

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for bad_alloc

16.1.33.2 Interfaces for Class bad_alloc

An LSB conforming implementation shall provide the generic methods for Class std::bad_alloc specified in Table 16-113, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-113 libstdcxx - Class bad_alloc Function Interfaces

bad_alloc::~bad_alloc()(GLIBCXX_3.4) [ISOCXX]
bad_alloc::~bad_alloc()(GLIBCXX_3.4) [ISOCXX]
bad_alloc::~bad_alloc()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::bad_alloc specified in Table 16-114, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-114 libstdcxx - Class bad_alloc Data Interfaces

typeinfo for bad_alloc(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for bad_alloc(GLIBCXX_3.4) [CXXABI-1.86]
vtable for bad_alloc(GLIBCXX_3.4) [CXXABI-1.86]

16.1.34 struct __numeric_limits_base

16.1.34.1 Interfaces for struct __numeric_limits_base

No external methods are defined for libstdcxx - struct __numeric_limits_base in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct __numeric_limits_base specified in Table 16-115, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-115 libstdcxx - struct __numeric_limits_base Data Interfaces

__numeric_limits_base::has_denorm(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::is_bounded(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::is_integer(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::round_style(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::has_infinity(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::max_exponent(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::min_exponent(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::is_specialized(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::max_exponent10(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::min_exponent10(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::tinyness_before(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::radix(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::traps(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::digits(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::digits10(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::is_exact(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::is_iec559(GLIBCXX_3.4) [ISOCXX]

__numeric_limits_base::is_modulo(GLIBCXX_3.4) [ISOCXX]
__numeric_limits_base::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.35 struct numeric_limits<long double>

16.1.35.1 Interfaces for struct numeric_limits<long double>

No external methods are defined for libstdcxx - struct numeric_limits<long double> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<long double> specified in Table 16-116, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-116 libstdcxx - struct numeric_limits<long double> Data Interfaces

numeric_limits<long double>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long double>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.36 struct numeric_limits<long long>

16.1.36.1 Interfaces for struct numeric_limits<long long>

No external methods are defined for libstdcxx - struct numeric_limits<long long> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<long long> specified in Table 16-117, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-117 libstdcxx - struct numeric_limits<long long> Data Interfaces

numeric_limits<long long>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long long>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.37 struct numeric_limits<unsigned long long>

16.1.37.1 Interfaces for struct numeric_limits<unsigned long long>

No external methods are defined for libstdcxx - struct numeric_limits<unsigned long long> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<unsigned long long> specified in Table 16-118, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-118 libstdcxx - struct numeric_limits<unsigned long long> Data Interfaces

numeric_limits<unsigned long long>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::digits10(GLIBCXX_3.4) [ISOCXX]

numeric_limits<unsigned long long>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long long>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.38 struct numeric_limits<float>

16.1.38.1 Interfaces for struct numeric_limits<float>

No external methods are defined for libstdcxx - struct numeric_limits<float> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<float> specified in Table 16-119, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-119 libstdcxx - struct numeric_limits<float> Data Interfaces

numeric_limits<float>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::is_iec559(GLIBCXX_3.4) [ISOCXX]

numeric_limits<float>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<float>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.39 struct numeric_limits<double>

16.1.39.1 Interfaces for struct numeric_limits<double>

No external methods are defined for libstdcxx - struct numeric_limits<double> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<double> specified in Table 16-120, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-120 libstdcxx - struct numeric_limits<double> Data Interfaces

numeric_limits<double>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<double>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.40 struct numeric_limits<short>

16.1.40.1 Interfaces for struct numeric_limits<short>

No external methods are defined for libstdcxx - struct numeric_limits<short> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<short> specified in Table 16-121, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-121 libstdcxx - struct numeric_limits<short> Data Interfaces

numeric_limits<short>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<short>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.41 struct numeric_limits<unsigned short>

16.1.41.1 Interfaces for struct numeric_limits<unsigned short>

No external methods are defined for libstdcxx - struct numeric_limits<unsigned short> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<unsigned short> specified in Table 16-122, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-122 libstdcxx - struct numeric_limits<unsigned short> Data Interfaces

numeric_limits<unsigned short>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned short>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.42 struct numeric_limits<int>

16.1.42.1 Interfaces for struct numeric_limits<int>

No external methods are defined for libstdcxx - struct numeric_limits<int> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<int> specified in Table 16-123, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-123 libstdcxx - struct numeric_limits<int> Data Interfaces

numeric_limits<int>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<int>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.43 struct numeric_limits<unsigned int>

16.1.43.1 Interfaces for struct numeric_limits<unsigned int>

No external methods are defined for libstdcxx - struct numeric_limits<unsigned int> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<unsigned int> specified in Table 16-124, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-124 libstdcxx - struct numeric_limits<unsigned int> Data Interfaces

numeric_limits<unsigned int>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned int>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.44 struct numeric_limits<long>

16.1.44.1 Interfaces for struct numeric_limits<long>

No external methods are defined for libstdcxx - struct numeric_limits<long> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<long> specified in Table 16-125, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-125 libstdcxx - struct numeric_limits<long> Data Interfaces

numeric_limits<long>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<long>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.45 struct numeric_limits<unsigned long>

16.1.45.1 Interfaces for struct numeric_limits<unsigned long>

No external methods are defined for libstdcxx - struct numeric_limits<unsigned long> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<unsigned long> specified in Table 16-126, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-126 libstdcxx - struct numeric_limits<unsigned long> Data Interfaces

numeric_limits<unsigned long>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned long>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.46 struct numeric_limits<wchar_t>

16.1.46.1 Interfaces for struct numeric_limits<wchar_t>

No external methods are defined for libstdcxx - struct numeric_limits<wchar_t> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<wchar_t> specified in Table 16-127, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-127 libstdcxx - struct numeric_limits<wchar_t> Data Interfaces

numeric_limits<wchar_t>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<wchar_t>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.47 struct numeric_limits<unsigned char>

16.1.47.1 Interfaces for struct numeric_limits<unsigned char>

No external methods are defined for libstdcxx - struct numeric_limits<unsigned char> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<unsigned char> specified in Table 16-128, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-128 libstdcxx - struct numeric_limits<unsigned char> Data Interfaces

numeric_limits<unsigned char>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<unsigned char>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.48 struct numeric_limits<signed char>

16.1.48.1 Interfaces for struct numeric_limits<signed char>

No external methods are defined for libstdcxx - struct numeric_limits<signed char> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<signed char> specified in Table 16-129, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-129 libstdcxx - struct numeric_limits<signed char> Data Interfaces

numeric_limits<signed char>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<signed char>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.49 struct numeric_limits<char>

16.1.49.1 Interfaces for struct numeric_limits<char>

No external methods are defined for libstdcxx - struct numeric_limits<char> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<char> specified in Table 16-130, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-130 libstdcxx - struct numeric_limits<char> Data Interfaces

numeric_limits<char>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<char>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.50 struct numeric_limits<bool>

16.1.50.1 Interfaces for struct numeric_limits<bool>

No external methods are defined for libstdcxx - struct numeric_limits<bool> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for struct numeric_limits<bool> specified in Table 16-131, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-131 libstdcxx - struct numeric_limits<bool> Data Interfaces

numeric_limits<bool>::has_denorm(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_bounded(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_integer(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::round_style(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::has_infinity(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::max_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::min_exponent(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::has_quiet_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_specialized(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::max_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::min_exponent10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::has_denorm_loss(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::tinyness_before(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::has_signaling_NaN(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::radix(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::traps(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::digits(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::digits10(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_exact(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_iec559(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_modulo(GLIBCXX_3.4) [ISOCXX]
numeric_limits<bool>::is_signed(GLIBCXX_3.4) [ISOCXX]

16.1.51 Class ctype_base

16.1.51.1 Class data for ctype_base

The Run Time Type Information for the std::ctype_base class is described by Table 16-132

Table 16-132 typeinfo for ctype_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for ctype_base

16.1.51.2 Interfaces for Class ctype_base

No external methods are defined for libstdcxx - Class std::ctype_base in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::ctype_base specified in Table 16-133, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-133 libstdcxx - Class ctype_base Data Interfaces

ctype_base::alnum(GLIBCXX_3.4) [ISOCXX]
ctype_base::alpha(GLIBCXX_3.4) [ISOCXX]
ctype_base::cntrl(GLIBCXX_3.4) [ISOCXX]
ctype_base::digit(GLIBCXX_3.4) [ISOCXX]
ctype_base::graph(GLIBCXX_3.4) [ISOCXX]
ctype_base::lower(GLIBCXX_3.4) [ISOCXX]
ctype_base::print(GLIBCXX_3.4) [ISOCXX]
ctype_base::punct(GLIBCXX_3.4) [ISOCXX]
ctype_base::space(GLIBCXX_3.4) [ISOCXX]
ctype_base::upper(GLIBCXX_3.4) [ISOCXX]
ctype_base::xdigit(GLIBCXX_3.4) [ISOCXX]
typeinfo for ctype_base(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ctype_base(GLIBCXX_3.4) [CXXABI-1.86]

16.1.52 Class __ctype_abstract_base<char>

16.1.52.1 Class data for __ctype_abstract_base<char>

The virtual table for the std::__ctype_abstract_base<char> class is described by Table 16-134

Table 16-134 Primary vtable for __ctype_abstract_base<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __ctype_abstract_base<char>
vfunc[0]:	NULL or __ctype_abstract_base<char>::~__ctype_abstract_base()
vfunc[1]:	NULL or __ctype_abstract_base<char>::~__ctype_abstract_base()
vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual
vfunc[9]:	__cxa_pure_virtual
vfunc[10]:	__cxa_pure_virtual
vfunc[11]:	__cxa_pure_virtual
vfunc[12]:	__cxa_pure_virtual
vfunc[13]:	__cxa_pure_virtual

16.1.52.2 Interfaces for Class __ctype_abstract_base<char>

No external methods are defined for libstdcxx - Class std::__ctype_abstract_base<char> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::__ctype_abstract_base<char> specified in Table 16-135, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-135 libstdcxx - Class __ctype_abstract_base<char> Data Interfaces

typeinfo for __ctype_abstract_base<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __ctype_abstract_base<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __ctype_abstract_base<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.53 Class __ctype_abstract_base<wchar_t>

16.1.53.1 Class data for __ctype_abstract_base<wchar_t>

The virtual table for the std::__ctype_abstract_base<wchar_t> class is described by Table 16-136

Table 16-136 Primary vtable for __ctype_abstract_base<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __ctype_abstract_base<wchar_t>
vfunc[0]:	NULL or __ctype_abstract_base<wchar_t>::~__ctype_abstract_base()
vfunc[1]:	NULL or __ctype_abstract_base<wchar_t>::~__ctype_abstract_base()
vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual
vfunc[9]:	__cxa_pure_virtual
vfunc[10]:	__cxa_pure_virtual
vfunc[11]:	__cxa_pure_virtual
vfunc[12]:	__cxa_pure_virtual
vfunc[13]:	__cxa_pure_virtual

16.1.53.2 Interfaces for Class __ctype_abstract_base<wchar_t>

No external methods are defined for libstdcxx - Class std::__ctype_abstract_base<wchar_t> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::__ctype_abstract_base<wchar_t> specified in Table 16-137, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-137 libstdcxx - Class __ctype_abstract_base<wchar_t> Data Interfaces

typeinfo for __ctype_abstract_base<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __ctype_abstract_base<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __ctype_abstract_base<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.54 Class ctype<char>

16.1.54.1 Class data for ctype<char>

The virtual table for the std::ctype<char> class is described by Table 16-138

Table 16-138 Primary vtable for ctype<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ctype<char>
vfunc[0]:	ctype<char>::~ctype()
vfunc[1]:	ctype<char>::~ctype()
vfunc[2]:	ctype<char>::do_toupper(char) const
vfunc[3]:	ctype<char>::do_toupper(char*, char const*) const
vfunc[4]:	ctype<char>::do_tolower(char) const
vfunc[5]:	ctype<char>::do_tolower(char*, char const*) const
vfunc[6]:	ctype<char>::do_widen(char) const
vfunc[7]:	ctype<char>::do_widen(char const*, char const*, char*) const
vfunc[8]:	ctype<char>::do_narrow(char, char) const
vfunc[9]:	ctype<char>::do_narrow(char const*, char const*, char, char*) const

16.1.54.2 Interfaces for Class ctype<char>

An LSB conforming implementation shall provide the generic methods for Class std::ctype<char> specified in Table 16-139, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-139 libstdcxx - Class ctype<char> Function Interfaces

ctype<char>::do_tolower(char*, char const*) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::do_tolower(char) const(GLIBCXX_3.4) [ISOCXX]

ctype<char>::do_toupper(char*, char const*) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::do_toupper(char) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::do_widen(char const*, char const*, char*) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::do_widen(char) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::do_narrow(char const*, char const*, char, char*) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::do_narrow(char, char) const(GLIBCXX_3.4) [ISOCXX]
ctype<char>::classic_table()(GLIBCXX_3.4) [ISOCXX]
ctype<char>::~ctype()(GLIBCXX_3.4) [ISOCXX]
ctype<char>::~ctype()(GLIBCXX_3.4) [ISOCXX]
ctype<char>::~ctype()(GLIBCXX_3.4) [ISOCXX]
bool has_facet<ctype<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::ctype<char> specified in Table 16-140, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-140 libstdcxx - Class ctype<char> Data Interfaces

ctype<char>::table_size(GLIBCXX_3.4) [ISOCXX]
ctype<char>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for ctype<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ctype<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for ctype<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.55 Class ctype<wchar_t>

16.1.55.1 Class data for ctype<wchar_t>

The virtual table for the std::ctype<wchar_t> class is described by Table 16-141

Table 16-141 Primary vtable for ctype<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ctype<wchar_t>
vfunc[0]:	ctype<wchar_t>::~ctype()
vfunc[1]:	ctype<wchar_t>::~ctype()
vfunc[2]:	ctype<wchar_t>::do_is(unsigned short, wchar_t) const

vfunc[3]:	ctype<wchar_t>::do_is(wchar_t const*, wchar_t const*, unsigned short*) const
vfunc[4]:	ctype<wchar_t>::do_scan_is(unsigned short, wchar_t const*, wchar_t const*) const
vfunc[5]:	ctype<wchar_t>::do_scan_not(unsigned short, wchar_t const*, wchar_t const*) const
vfunc[6]:	ctype<wchar_t>::do_toupper(wchar_t) const
vfunc[7]:	ctype<wchar_t>::do_toupper(wchar_t*, wchar_t const*) const
vfunc[8]:	ctype<wchar_t>::do_tolower(wchar_t) const
vfunc[9]:	ctype<wchar_t>::do_tolower(wchar_t*, wchar_t const*) const
vfunc[10]:	ctype<wchar_t>::do_widen(char) const
vfunc[11]:	ctype<wchar_t>::do_widen(char const*, char const*, wchar_t*) const
vfunc[12]:	ctype<wchar_t>::do_narrow(wchar_t, char) const
vfunc[13]:	ctype<wchar_t>::do_narrow(wchar_t const*, wchar_t const*, char, char*) const

The Run Time Type Information for the std::ctype<wchar_t> class is described by Table 16-142

Table 16-142 typeinfo for ctype<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for ctype<wchar_t>

16.1.55.2 Interfaces for Class ctype<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::ctype<wchar_t> specified in Table 16-143, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-143 libstdcxx - Class ctype<wchar_t> Function Interfaces

ctype<wchar_t>::do_scan_is(unsigned short, wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_tolower(wchar_t*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_tolower(wchar_t) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_toupper(wchar_t*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_toupper(wchar_t) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_scan_not(unsigned short, wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::_M_convert_to_wmask(unsigned short) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_is(wchar_t const*, wchar_t const*, unsigned short*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_is(unsigned short, wchar_t) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_widen(char const*, char const*, wchar_t*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_widen(char) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_narrow(wchar_t const*, wchar_t const*, char, char*) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::do_narrow(wchar_t, char) const(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::_M_initialize_ctype()(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::~ctype()(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::~ctype()(GLIBCXX_3.4) [ISOCXX]
ctype<wchar_t>::~ctype()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::ctype<wchar_t> specified in Table 16-144, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-144 libstdcxx - Class ctype<wchar_t> Data Interfaces

ctype<wchar_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for ctype<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ctype<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for ctype<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.56 Class ctype_byname<char>

16.1.56.1 Class data for ctype_byname<char>

The virtual table for the std::ctype_byname<char> class is described by Table 16-145

Table 16-145 Primary vtable for ctype_byname<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ctype_byname<char>
vfunc[0]:	ctype_byname<char>::~ctype_byname()
vfunc[1]:	ctype_byname<char>::~ctype_byname()
vfunc[2]:	ctype<char>::do_toupper(char) const
vfunc[3]:	ctype<char>::do_toupper(char*, char const*) const
vfunc[4]:	ctype<char>::do_tolower(char) const
vfunc[5]:	ctype<char>::do_tolower(char*, char const*) const
vfunc[6]:	ctype<char>::do_widen(char) const
vfunc[7]:	ctype<char>::do_widen(char const*, char const*, char*) const
vfunc[8]:	ctype<char>::do_narrow(char, char) const
vfunc[9]:	ctype<char>::do_narrow(char const*, char const*, char, char*) const

The Run Time Type Information for the std::ctype_byname<char> class is described by Table 16-146

Table 16-146 typeinfo for ctype_byname<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for ctype_byname<char>

16.1.56.2 Interfaces for Class ctype_byname<char>

An LSB conforming implementation shall provide the generic methods for Class std::ctype_byname<char> specified in Table 16-147, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-147 libstdcxx - Class ctype_byname<char> Function Interfaces

ctype_byname<char>::~ctype_byname()(GLIBCXX_3.4) [ISOCXX]
ctype_byname<char>::~ctype_byname()(GLIBCXX_3.4) [ISOCXX]
ctype_byname<char>::~ctype_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::ctype_byname<char> specified in Table 16-148, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-148 libstdcxx - Class ctype_byname<char> Data Interfaces

typeinfo for ctype_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ctype_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for ctype_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.57 Class ctype_byname<wchar_t>

16.1.57.1 Class data for ctype_byname<wchar_t>

The virtual table for the std::ctype_byname<wchar_t> class is described by Table 16-149

Table 16-149 Primary vtable for ctype_byname<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ctype_byname<wchar_t>
vfunc[0]:	ctype_byname<wchar_t>::~ctype_byname()
vfunc[1]:	ctype_byname<wchar_t>::~ctype_byname()
vfunc[2]:	ctype<wchar_t>::do_is(unsigned short, wchar_t) const
vfunc[3]:	ctype<wchar_t>::do_is(wchar_t const*, wchar_t const*, unsigned short*) const
vfunc[4]:	ctype<wchar_t>::do_scan_is(unsigned short, wchar_t const*, wchar_t const*) const
vfunc[5]:	ctype<wchar_t>::do_scan_not(unsigned short, wchar_t const*, wchar_t const*) const
vfunc[6]:	ctype<wchar_t>::do_toupper(wchar_t) const

vfunc[7]:	ctype<wchar_t>::do_toupper(wchar_t *, wchar_t const*) const
vfunc[8]:	ctype<wchar_t>::do_tolower(wchar_t) const
vfunc[9]:	ctype<wchar_t>::do_tolower(wchar_t *, wchar_t const*) const
vfunc[10]:	ctype<wchar_t>::do_widen(char) const
vfunc[11]:	ctype<wchar_t>::do_widen(char const*, char const*, wchar_t*) const
vfunc[12]:	ctype<wchar_t>::do_narrow(wchar_t, char) const
vfunc[13]:	ctype<wchar_t>::do_narrow(wchar_t const*, wchar_t const*, char, char*) const

The Run Time Type Information for the std::ctype_byname<wchar_t> class is described by Table 16-150

Table 16-150 typeinfo for ctype_byname<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for ctype_byname<wchar_t>

16.1.57.2 Interfaces for Class ctype_byname<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::ctype_byname<wchar_t> specified in Table 16-151, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-151 libstdcxx - Class ctype_byname<wchar_t> Function Interfaces

ctype_byname<wchar_t>::~ctype_byname()(GLIBCXX_3.4) [ISOCXX]
ctype_byname<wchar_t>::~ctype_byname()(GLIBCXX_3.4) [ISOCXX]
ctype_byname<wchar_t>::~ctype_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::ctype_byname<wchar_t> specified in Table 16-152, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-152 libstdcxx - Class ctype_byname<wchar_t> Data Interfaces

typeinfo for ctype_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ctype_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

vtable for ctype_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.58 Class basic_string<char, char_traits<char>, allocator<char> >

16.1.58.1 Interfaces for Class basic_string<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_string<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-153, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-153 libstdcxx - Class basic_string<char, char_traits<char>, allocator<char> > Function Interfaces

basic_string<char, char_traits<char>, allocator<char> >::_M_disjunct(char const*) const(GLIBCXX_3.4.5) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::get_allocator() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::end() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_is_leaked() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_is_shared() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::data() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::rend() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::size() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::begin() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::c_str() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::empty() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_M_rep() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::length() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::rbegin() const(GLIBCXX_3.4) [ISOCXX]

basic_string<char, char_traits<char>, allocator<char> >::_M_data() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_M_iend() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::compare(char const*) const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::compare(basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::capacity() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::max_size() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_M_ibegin() const(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Alloc_hider::_Alloc_hider(char*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Alloc_hider::_Alloc_hider(char*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_M_leak_hard()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_S_empty_rep()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_S_copy_chars(char*, __gnu_cxx::__normal_iterator<char const*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char const*, basic_string<char, char_traits<char>, allocator<char> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_S_copy_chars(char*, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_S_copy_chars(char*, char const*, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_S_copy_chars(char*, char*, char*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::end()(GLIBCXX_3.4) [ISOCXX]

basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_destroy(allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_dispose(allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_refcopy()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_refdata()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_S_empty_rep()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_set_leaked()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_set_sharable()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_M_grab(allocator<char> const&, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::rend()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::swap(basic_string<char, char_traits<char>, allocator<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::begin()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::clear()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::erase(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::erase(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::append(char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::append(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::assign(char const*)(GLIBCXX_3.4) [ISOCXX]

basic_string<char, char_traits<char>, allocator<char> >::assign(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::insert(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::rbegin()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_M_data(char*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_M_leak()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::replace(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char const*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char const*, basic_string<char, char_traits<char>, allocator<char> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::replace(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::replace(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, char const*, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::replace(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::replace(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, char*, char*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::replace(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >,

__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::push_back(char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string(char const*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string(allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string<__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > > >(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string<char const*>(char const*, char const*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string<char*>(char*, char*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string(char const*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string(allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string<__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > > >(__gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, __gnu_cxx::__normal_iterator<char*, basic_string<char, char_traits<char>, allocator<char> > >, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]

basic_string<char, char_traits<char>, allocator<char> >::basic_string<char const*>(char const*, char const*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::basic_string<char*>(char*, char*, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::~basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::~basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::operator=(char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::operator=(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::operator=(char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::operator+=(char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::operator+=(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::operator+=(char)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_string<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-154, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-154 libstdcxx - Class basic_string<char, char_traits<char>, allocator<char> > Data Interfaces

basic_string<char, char_traits<char>, allocator<char> >::_Rep::_S_max_size(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_S_terminal(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::_Rep::_S_empty_rep_storage(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::npos(GLIBCXX_3.4) [ISOCXX]

16.1.59 Class basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

16.1.59.1 Interfaces for Class basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-155, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-155 libstdcxx - Class basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_disjunct(wchar_t const*) const(GLIBCXX_3.4.5) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::get_allocator() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::end() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_is_leaked() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_is_shared() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::data() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rend() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::size() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::begin() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::c_str() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::empty() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_rep() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::length() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rbegin() const(GLIBCXX_3.4) [ISOCXX]

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_data() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_iend() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::compare(wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::compare(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::capacity() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::max_size() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_ibegin() const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Alloc_hider::_Alloc_hider(wchar_t*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Alloc_hider::_Alloc_hider(wchar_t*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_leak_hard()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_S_empty_rep()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_S_copy_chars(wchar_t*, __gnu_cxx::__normal_iterator<wchar_t const*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t const*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_S_copy_chars(wchar_t*, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_S_copy_chars(wchar_t*, wchar_t const*, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_S_copy_chars(wchar_t*, wchar_t*, wchar_t*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::end()(GLIBCXX_3.4) [ISOCXX]

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_destroy(allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_dispose(allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_refcopy()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_refdata()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_S_empty_rep()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_set_leaked()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_set_sharable()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_M_grab(allocator<wchar_t> const&, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rend()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::swap(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::begin()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::clear()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::erase(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::erase(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::append(wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::append(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::assign(wchar_t const*)(GLIBCXX_3.4) [ISOCXX]

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::assign(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::insert(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rbegin()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_data(wchar_t*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_leak()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t const*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t const*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, wchar_t const*, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t,

char_traits<wchar_t>, allocator<wchar_t> > >, wchar_t*, wchar_t*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::push_back(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string(wchar_t const*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string(allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string<__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > > >(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string<wchar_t const*>(wchar_t const*, wchar_t const*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string<wchar_t*>(wchar_t*, wchar_t*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string(wchar_t const*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string(allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string<__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > > >(__gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, __gnu_cxx::__normal_iterator<wchar_t*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > >, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string<wchar_t const*>(wchar_t const*, wchar_t const*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_string<wchar_t*>(wchar_t*, wchar_t*, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_string()(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::operator=(wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::operator=(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::operator=(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::operator+=(wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::operator+=(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::operator+=(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> > operator+<char, char_traits<char>, allocator<char> >(char const*, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> > operator+<char, char_traits<char>, allocator<char> >(basic_string<char, char_traits<char>,

allocator<char> > const&, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> > operator+<char, char_traits<char>, allocator<char> >(char, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > operator+<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(wchar_t const*, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > operator+<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > operator+<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(wchar_t, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-156, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-156 libstdcxx - Class basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Data Interfaces

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_S_max_size(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_S_terminal(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_Rep::_S_empty_rep_storage(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::npos(GLIBCXX_3.4) [ISOCXX]

16.1.60 Class basic_stringstream<char, char_traits<char>, allocator<char> >

16.1.60.1 Class data for basic_stringstream<char, char_traits<char>, allocator<char> >

The virtual table for the std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> > class is described by Table 16-157

Table 16-157 VTT for basic_stringstream<char, char_traits<char>, allocator<char> >

VTT Name	_ZTTSt18basic_stringstreamIcSt11char_traitsIcESaIcEE
Number of Entries	10

16.1.60.2 Interfaces for Class basic_stringstream<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-158, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-158 libstdcxx - Class basic_stringstream<char, char_traits<char>, allocator<char> > Function Interfaces

basic_stringstream<char, char_traits<char>, allocator<char> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::str(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::basic_stringstream(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::basic_stringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::basic_stringstream(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::basic_stringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-159, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-159 libstdcxx - Class basic_stringstream<char, char_traits<char>, allocator<char> > Data Interfaces

typeinfo for basic_stringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_stringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_stringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_stringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.61 Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

16.1.61.1 Class data for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

The virtual table for the std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 16-160

Table 16-160 VTT for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

VTT Name	_ZTTSt18basic_stringstreamIwSt11char_traitsIwESaIwEE
Number of Entries	10

16.1.61.2 Interfaces for Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-161, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-161 libstdcxx - Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces

basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]

basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringstream(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringstream(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]
basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-162, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-162 libstdcxx - Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Data Interfaces

typeinfo for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.62 Class basic_istringstream<char, char_traits<char>, allocator<char> >

16.1.62.1 Class data for basic_istringstream<char, char_traits<char>, allocator<char> >

The virtual table for the std::basic_istringstream<char, std::char_traits<char>, std::allocator<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_istringstream<char, std::char_traits<char>, std::allocator<char> > class is described by Table 16-163

Table 16-163 VTT for basic_istringstream<char, char_traits<char>, allocator<char> >

VTT Name	_ZTTSt19basic_istringstreamIcSt11char_traitsIcESaIcEE
Number of Entries	4

16.1.62.2 Interfaces for Class basic_istringstream<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_istringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-164, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-164 libstdcxx - Class basic_istringstream<char, char_traits<char>, allocator<char> > Function Interfaces

basic_istringstream<char, char_traits<char>, allocator<char> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::str(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::basic_istringstream(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::basic_istringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::basic_istringstream(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::basic_istringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]

basic_istringstream<char, char_traits<char>, allocator<char> >::~basic_istringstream()(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::~basic_istringstream()(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<char, char_traits<char>, allocator<char> >::~basic_istringstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_istringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-165, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-165 libstdcxx - Class basic_istringstream<char, char_traits<char>, allocator<char> > Data Interfaces

typeinfo for basic_istringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_istringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_istringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_istringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.63 Class basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

16.1.63.1 Class data for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

The virtual table for the std::basic_istringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_istringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 16-166

Table 16-166 VTT for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

VTT Name	_ZTTSt19basic_istringstreamIwSt11char_traitsIwESaIwEE
Number of Entries	4

16.1.63.2 Interfaces for Class basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_istringstream<wchar_t, std::char_traits<wchar_t>,

std::allocator<wchar_t> > specified in Table 16-167, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-167 libstdcxx - Class basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces

basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_istringstream(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_istringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_istringstream(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_istringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_istringstream()(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_istringstream()(GLIBCXX_3.4) [ISOCXX]
basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_istringstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_istringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-168, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-168 libstdcxx - Class basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Data Interfaces

typeinfo for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

vtable for basic_istringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.64 Class basic_ostringstream<char, char_traits<char>, allocator<char> >

16.1.64.1 Class data for basic_ostringstream<char, char_traits<char>, allocator<char> >

The virtual table for the std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> > class is described by Table 16-169

Table 16-169 VTT for basic_ostringstream<char, char_traits<char>, allocator<char> >

VTT Name	_ZTTSt19basic_ostringstreamIcSt11char_traitsIcESaIcEE
Number of Entries	4

16.1.64.2 Interfaces for Class basic_ostringstream<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-170, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-170 libstdcxx - Class basic_ostringstream<char, char_traits<char>, allocator<char> > Function Interfaces

basic_ostringstream<char, char_traits<char>, allocator<char> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::str(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::basic_ostringstream(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::basic_ostringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::basic_ostringstream(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]

basic_ostringstream<char, char_traits<char>, allocator<char> >::basic_ostringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<char, char_traits<char>, allocator<char> >::~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-171, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-171 libstdcxx - Class basic_ostringstream<char, char_traits<char>, allocator<char> > Data Interfaces

typeinfo for basic_ostringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ostringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ostringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ostringstream<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.65 Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

16.1.65.1 Class data for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

The virtual table for the std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 16-172

Table 16-172 VTT for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

VTT Name	_ZTTSt19basic_ostringstreamIwSt11char_traitsIwESaIwEE
Number of Entries	4

16.1.65.2 Interfaces for Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-173, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-173 libstdcxx - Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces

basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_ostringstream(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_ostringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_ostringstream(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_ostringstream(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]
basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-174, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-174 libstdcxx - Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Data Interfaces

typeinfo for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.66 Class basic_stringbuf<char, char_traits<char>, allocator<char> >

16.1.66.1 Class data for basic_stringbuf<char, char_traits<char>, allocator<char> >

The virtual table for the std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char> > class is described by Table 16-175

Table 16-175 Primary vtable for basic_stringbuf<char, char_traits<char>, allocator<char> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_stringbuf<char, char_traits<char>, allocator<char> >
vfunc[0]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::~basic_stringbuf()
vfunc[1]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::~basic_stringbuf()
vfunc[2]:	basic_streambuf<char, char_traits<char> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<char, char_traits<char> >::sync()
vfunc[7]:	basic_streambuf<char, char_traits<char> >::showmanyc()
vfunc[8]:	See architecture specific part.

vfunc[9]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::underflow()
vfunc[10]:	basic_streambuf<char, char_traits<char> >::uflow()
vfunc[11]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::pbackfail(int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	basic_stringbuf<char, char_traits<char>, allocator<char> >::overflow(int)

The Run Time Type Information for the std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char> > class is described by Table 16-176

Table 16-176 typeinfo for basic_stringbuf<char, char_traits<char>, allocator<char> >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for basic_stringbuf<char, char_traits<char>, allocator<char> >

16.1.66.2 Interfaces for Class basic_stringbuf<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-177, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-177 libstdcxx - Class basic_stringbuf<char, char_traits<char>, allocator<char> > Function Interfaces

basic_stringbuf<char, char_traits<char>, allocator<char> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::_M_update_egptr()(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::_M_stringbuf_init(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::str(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]

basic_stringbuf<char, char_traits<char>, allocator<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::overflow(int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::pbackfail(int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::showmanyc()(GLIBCXX_3.4.6) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::underflow()(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::basic_stringbuf(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::basic_stringbuf(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::basic_stringbuf(basic_string<char, char_traits<char>, allocator<char> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::basic_stringbuf(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::~basic_stringbuf()(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<char, char_traits<char>, allocator<char> >::~basic_stringbuf()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char> > specified in Table 16-178, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-178 libstdcxx - Class basic_stringbuf<char, char_traits<char>, allocator<char> > Data Interfaces

typeinfo for basic_stringbuf<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_stringbuf<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_stringbuf<char, char_traits<char>, allocator<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.67 Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

16.1.67.1 Class data for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

The virtual table for the std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 16-179

Table 16-179 Primary vtable for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringbuf()
vfunc[1]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringbuf()
vfunc[2]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::underflow()

vfunc[10]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::overflow(unsigned int)

The Run Time Type Information for the std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 16-180

Table 16-180 typeinfo for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

16.1.67.2 Interfaces for Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-181, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-181 libstdcxx - Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces

basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str() const(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_update_egptr()(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_stringbuf_init(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::str(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]

basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::overflow(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::pbackfail(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::showmanyc()(GLIBCXX_3.4.6) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::underflow()(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringbuf(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringbuf(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringbuf(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::basic_stringbuf(_Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringbuf()(GLIBCXX_3.4) [ISOCXX]
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringbuf()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 16-182, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-182 libstdcxx - Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Data Interfaces

typeinfo for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.68 Class basic_iostream<char, char_traits<char> >

16.1.68.1 Class data for basic_iostream<char, char_traits<char> >

The virtual table for the std::basic_iostream<char, std::char_traits<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_iostream<char, std::char_traits<char> > class is described by Table 16-183

Table 16-183 VTT for basic_iostream<char, char_traits<char> >

VTT Name	_ZTTSd
Number of Entries	7

16.1.68.2 Interfaces for Class basic_iostream<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_iostream<char, std::char_traits<char> > specified in Table 16-184, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-184 libstdcxx - Class basic_iostream<char, char_traits<char> > Function Interfaces

basic_iostream<char, char_traits<char> >::basic_iostream(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_iostream<char, char_traits<char> >::basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<char, char_traits<char> >::basic_iostream(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_iostream<char, char_traits<char> >::basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<char, char_traits<char> >::~basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<char, char_traits<char> >::~basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<char, char_traits<char> >::~basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char_traits<char> >(basic_istream<char, char_traits<char> >&, signed char*)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_iostream<char, std::char_traits<char> > specified in Table 16-185, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-185 libstdcxx - Class basic_iostream<char, char_traits<char> > Data Interfaces

typeinfo for basic_iostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_iostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_iostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_iostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.69 Class basic_iostream<wchar_t, char_traits<wchar_t> >

16.1.69.1 Class data for basic_iostream<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_iostream<wchar_t, std::char_traits<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_iostream<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-186

Table 16-186 VTT for basic_iostream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt14basic_iostreamIwSt11char_traitsIwEE
Number of Entries	7

16.1.69.2 Interfaces for Class basic_iostream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_iostream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-187, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-187 libstdcxx - Class basic_iostream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_iostream<wchar_t, char_traits<wchar_t> >::basic_iostream(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]

basic_iostream<wchar_t, char_traits<wchar_t> >::basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<wchar_t, char_traits<wchar_t> >::basic_iostream(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_iostream<wchar_t, char_traits<wchar_t> >::basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [ISOCXX]
basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_iostream<wchar_t, std::char_traits<wchar_t>> specified in Table 16-188, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-188 libstdcxx - Class basic_iostream<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_iostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_iostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_iostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_iostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.70 Class basic_istream<char, char_traits<char> >

16.1.70.1 Class data for basic_istream<char, char_traits<char> >

The virtual table for the std::basic_istream<char, std::char_traits<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_istream<char, std::char_traits<char> > class is described by Table 16-189

Table 16-189 VTT for basic_istream<char, char_traits<char> >

VTT Name	_ZTTSi
Number of Entries	2

16.1.70.2 Interfaces for Class basic_istream<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_istream<char, std::char_traits<char> > specified in Table 16-190, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-190 libstdcxx - Class basic_istream<char, char_traits<char> > Function Interfaces

basic_istream<char, char_traits<char> >::gcount() const(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::sentry::operator bool() const(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::get(basic_streambuf<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::get(basic_streambuf<char, char_traits<char> >&, char)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::get(char&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::get()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::peek()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::sync()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::seekg(fpos<__mbstate_t>)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::tellg()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::unget()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::ignore()(GLIBCXX_3.4.5) [ISOCXX]
basic_istream<char, char_traits<char> >::sentry::sentry(basic_istream<char, char_traits<char> >&, bool)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::sentry::sentry(basic_istream<char, char_traits<char> >&, bool)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::putback(char)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::basic_istream(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::basic_istream(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]

basic_istream<char, char_traits<char> >::basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::~basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::~basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::~basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(basic_istream<char, char_traits<char> >& (*)(basic_istream<char, char_traits<char> >&))(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(ios_base& (*)(ios_base&))(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(basic_ios<char, char_traits<char> >& (*)(basic_ios<char, char_traits<char> >&))(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(void*&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(bool&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(double&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(long double&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(float&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(int&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(unsigned int&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(unsigned long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(short&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(unsigned short&)(GLIBCXX_3.4) [ISOCXX]

basic_istream<char, char_traits<char> >::operator>>(long long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >::operator>>(unsigned long long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& ws<char, char_traits<char> >(basic_istream<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& getline<char, char_traits<char>, allocator<char> >(basic_istream<char, char_traits<char> >&, basic_string<char, char_traits<char>, allocator<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& getline<char, char_traits<char>, allocator<char> >(basic_istream<char, char_traits<char> >&, basic_string<char, char_traits<char>, allocator<char> >&, char)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char_traits<char> >(basic_istream<char, char_traits<char> >&, unsigned char*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char_traits<char> >(basic_istream<char, char_traits<char> >&, signed char&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char_traits<char> >(basic_istream<char, char_traits<char> >&, unsigned char&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, char*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, char&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, _Setiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, _Setprecision)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, _Resetiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, _Setw)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, _Setbase)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><char, char_traits<char> >(basic_istream<char, char_traits<char> >&, _Setfill<char>)(GLIBCXX_3.4) [ISOCXX]

basic_istream<char, char_traits<char> >& operator>><char, char_traits<char>, allocator<char> >(basic_istream<char, char_traits<char> >&, basic_string<char, char_traits<char>, allocator<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><double, char, char_traits<char> >(basic_istream<char, char_traits<char> >&, complex<double>&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><long double, char, char_traits<char> >(basic_istream<char, char_traits<char> >&, complex<long double>&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<char, char_traits<char> >& operator>><float, char, char_traits<char> >(basic_istream<char, char_traits<char> >&, complex<float>&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_istream<char, std::char_traits<char> > specified in Table 16-191, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-191 libstdcxx - Class basic_istream<char, char_traits<char> > Data Interfaces

typeinfo for basic_istream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_istream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_istream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_istream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.71 Class basic_istream<wchar_t, char_traits<wchar_t> >

16.1.71.1 Class data for basic_istream<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_istream<wchar_t, std::char_traits<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_istream<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-192

Table 16-192 VTT for basic_istream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt13basic_istreamIwSt11char_traitsIwEE
Number of Entries	2

16.1.71.2 Interfaces for Class basic_istream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_istream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-193, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-193 libstdcxx - Class basic_istream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_istream<wchar_t, char_traits<wchar_t> >::gcount() const(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::sentry::operator bool() const(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::get(basic_streambuf<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::get(basic_streambuf<wchar_t, char_traits<wchar_t> >&, wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::get(wchar_t&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::get()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::peek()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::sync()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::seekg(fpos<__mbstate_t>)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::tellg()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::unget()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::ignore()(GLIBCXX_3.4.5) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::sentry::sentry(basic_istream<wchar_t, char_traits<wchar_t> >&, bool)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::sentry::sentry(basic_istream<wchar_t, char_traits<wchar_t> >&, bool)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::putback(wchar_t)(GLIBCXX_3.4) [ISOCXX]

basic_istream<wchar_t, char_traits<wchar_t> >::basic_istream(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::basic_istream(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(basic_istream<wchar_t, char_traits<wchar_t> >& (*)(basic_istream<wchar_t, char_traits<wchar_t> >&))(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(ios_base& (*)(ios_base&))(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(basic_ios<wchar_t, char_traits<wchar_t> >& (*)(basic_ios<wchar_t, char_traits<wchar_t> >&))(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(void*&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(bool&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(double&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(long double&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(float&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(int&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(unsigned int&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(long&)(GLIBCXX_3.4) [ISOCXX]

basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(unsigned long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(short&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(unsigned short&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(long long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::operator>>(unsigned long long&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& ws<wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& getline<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& getline<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&, wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><double, wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, complex<double>&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><long double, wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, complex<long double>&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><float, wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, complex<float>&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, wchar_t*)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, wchar_t&)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, _Setiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, _Setprecision)(GLIBCXX_3.4) [ISOCXX]

basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, _Resetiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, _Setw)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, _Setbase)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, _Setfill<wchar_t>)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >& operator>><wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(basic_istream<wchar_t, char_traits<wchar_t> >&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_istream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-194, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-194 libstdcxx - Class basic_istream<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_istream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_istream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_istream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_istream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.72 Class istreambuf_iterator<wchar_t, char_traits<wchar_t> >

16.1.72.1 Interfaces for Class istreambuf_iterator<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > specified in Table 16-195, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-195 libstdcxx - Class istreambuf_iterator<wchar_t, char_traits<wchar_t> > Function Interfaces

istreambuf_iterator<wchar_t, char_traits<wchar_t> >::operator++()(GLIBCXX_3.4.5) [ISOCXX]

16.1.73 Class istreambuf_iterator<char, char_traits<char> >

16.1.73.1 Interfaces for Class istreambuf_iterator<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::istreambuf_iterator<char, std::char_traits<char> > specified in Table 16-196, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-196 libstdcxx - Class istreambuf_iterator<char, char_traits<char> > Function Interfaces

istreambuf_iterator<char, char_traits<char> >::operator++()(GLIBCXX_3.4.5) [ISOCXX]

16.1.74 Class basic_ostream<char, char_traits<char> >

16.1.74.1 Class data for basic_ostream<char, char_traits<char> >

The virtual table for the std::basic_ostream<char, std::char_traits<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ostream<char, std::char_traits<char> > class is described by Table 16-197

Table 16-197 VTT for basic_ostream<char, char_traits<char> >

VTT Name	_ZTTSo
Number of Entries	2

16.1.74.2 Interfaces for Class basic_ostream<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ostream<char, std::char_traits<char> > specified in Table 16-198, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-198 libstdcxx - Class basic_ostream<char, char_traits<char> > Function Interfaces

basic_ostream<char, char_traits<char> >::sentry::operator bool() const(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::put(char)(GLIBCXX_3.4) [ISOCXX]

basic_ostream<char, char_traits<char> >::flush()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::seekp(fpos<__mbstate_t>)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::tellp()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::sentry::sentry(basic_ostream<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::sentry::sentry(basic_ostream<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::sentry::~sentry()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::sentry::~sentry()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::basic_ostream(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::basic_ostream(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(basic_ostream<char, char_traits<char> >& (*)(basic_ostream<char, char_traits<char> >&))(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(ios_base& (*)(ios_base&))(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(basic_ios<char, char_traits<char> >& (*)(basic_ios<char, char_traits<char> >&))(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(void const*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]

basic_ostream<char, char_traits<char> >::operator<<(bool)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(double)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(long double)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(float)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(int)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(short)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(unsigned short)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(long long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::operator<<(unsigned long long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& endl<char, char_traits<char> >(basic_ostream<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& ends<char, char_traits<char> >(basic_ostream<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& flush<char, char_traits<char> >(basic_ostream<char, char_traits<char> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char_traits<char> >(basic_ostream<char, char_traits<char> >&, signed char const*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char_traits<char> >(basic_ostream<char, char_traits<char> >&, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char_traits<char> >(basic_ostream<char, char_traits<char> >&, unsigned char const*)(GLIBCXX_3.4) [ISOCXX]

basic_ostream<char, char_traits<char> >& operator<< <char_traits<char> >(basic_ostream<char, char_traits<char> >&, signed char)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char_traits<char> >(basic_ostream<char, char_traits<char> >&, char)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char_traits<char> >(basic_ostream<char, char_traits<char> >&, unsigned char)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, _Setiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, _Setprecision)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, _Resetiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, _Setw)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, _Setbase)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, _Setfill<char>)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <char, char_traits<char>, allocator<char> >(basic_ostream<char, char_traits<char> >&, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <double, char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, complex<double> const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <long double, char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, complex<long double> const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >& operator<< <float, char, char_traits<char> >(basic_ostream<char, char_traits<char> >&, complex<float> const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ostream<char, std::char_traits<char> > specified in Table 16-199, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-199 libstdcxx - Class basic_ostream<char, char_traits<char> > Data Interfaces

typeinfo for basic_ostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ostream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.75 Class basic_ostream<wchar_t, char_traits<wchar_t> >

16.1.75.1 Class data for basic_ostream<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_ostream<wchar_t, std::char_traits<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ostream<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-200

Table 16-200 VTT for basic_ostream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt13basic_ostreamIwSt11char_traitsIwEE
Number of Entries	2

16.1.75.2 Interfaces for Class basic_ostream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ostream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-201, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-201 libstdcxx - Class basic_ostream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_ostream<wchar_t, char_traits<wchar_t> >::sentry::operator bool() const(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::put(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::flush()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::seekp(fpos<__mbstate_t>)(GLIBCXX_3.4) [ISOCXX]

basic_ostream<wchar_t, char_traits<wchar_t> >::tellp()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::sentry::sentry(basic_ostream<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::sentry::sentry(basic_ostream<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::sentry::~sentry()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::sentry::~sentry()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::basic_ostream(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::basic_ostream(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(basic_ostream<wchar_t, char_traits<wchar_t> >& (*)(basic_ostream<wchar_t, char_traits<wchar_t> >&))(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(ios_base& (*)(ios_base&))(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(basic_ios<wchar_t, char_traits<wchar_t> >& (*)(basic_ios<wchar_t, char_traits<wchar_t> >&))(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(void const*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(bool)(GLIBCXX_3.4) [ISOCXX]

basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(double)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(long double)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(float)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(int)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(short)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(unsigned short)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(long long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::operator<<(unsigned long long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& endl<wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& ends<wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& flush<wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <double, wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, complex<double> const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <long double, wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, complex<long double> const&)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <float, wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, complex<float> const&)(GLIBCXX_3.4) [ISOCXX]

basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, _Setiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, _Setprecision)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, _Resetiosflags)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, _Setw)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, _Setbase)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, _Setfill<wchar_t>)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, char)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >& operator<< <wchar_t, char_traits<wchar_t>, allocator<wchar_t> >(basic_ostream<wchar_t, char_traits<wchar_t> >&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ostream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-202, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-202 libstdcxx - Class basic_ostream<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_ostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for basic_ostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ostream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.76 Class basic_fstream<char, char_traits<char> >

16.1.76.1 Class data for basic_fstream<char, char_traits<char> >

The virtual table for the std::basic_fstream<char, std::char_traits<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_fstream<char, std::char_traits<char> > class is described by Table 16-203

Table 16-203 VTT for basic_fstream<char, char_traits<char> >

VTT Name	_ZTTSt13basic_fstreamIcSt11char_traitsIcEE
Number of Entries	10

16.1.76.2 Interfaces for Class basic_fstream<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_fstream<char, std::char_traits<char> > specified in Table 16-204, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-204 libstdcxx - Class basic_fstream<char, char_traits<char> > Function Interfaces

basic_fstream<char, char_traits<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::is_open() const(GLIBCXX_3.4.5) [ISOCXX]
basic_fstream<char, char_traits<char> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::is_open()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::basic_fstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::basic_fstream()(GLIBCXX_3.4) [ISOCXX]

basic_fstream<char, char_traits<char> >::basic_fstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<char, char_traits<char> >::~basic_fstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_fstream<char, std::char_traits<char> > specified in Table 16-205, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-205 libstdcxx - Class basic_fstream<char, char_traits<char> > Data Interfaces

typeinfo for basic_fstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_fstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_fstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_fstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.77 Class basic_fstream<wchar_t, char_traits<wchar_t> >

16.1.77.1 Class data for basic_fstream<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_fstream<wchar_t, std::char_traits<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_fstream<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-206

Table 16-206 VTT for basic_fstream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt13basic_fstreamIwSt11char_traitsIwEE
Number of Entries	10

16.1.77.2 Interfaces for Class basic_fstream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_fstream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-207, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-207 libstdcxx - Class basic_fstream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_fstream<wchar_t, char_traits<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::is_open() const(GLIBCXX_3.4.5) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::is_open()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::basic_fstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::basic_fstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()(GLIBCXX_3.4) [ISOCXX]
basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_fstream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-208, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-208 libstdcxx - Class basic_fstream<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_fstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_fstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_fstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_fstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.78 Class basic_ifstream<char, char_traits<char> >

16.1.78.1 Class data for basic_ifstream<char, char_traits<char> >

The virtual table for the std::basic_ifstream<char, std::char_traits<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ifstream<char, std::char_traits<char> > class is described by Table 16-209

Table 16-209 VTT for basic_ifstream<char, char_traits<char> >

VTT Name	_ZTTSt14basic_ifstreamIcSt11char_traitsIcEE
Number of Entries	4

16.1.78.2 Interfaces for Class basic_ifstream<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ifstream<char, std::char_traits<char> > specified in Table 16-210, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-210 libstdcxx - Class basic_ifstream<char, char_traits<char> > Function Interfaces

basic_ifstream<char, char_traits<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::is_open() const(GLIBCXX_3.4.5) [ISOCXX]
basic_ifstream<char, char_traits<char> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::is_open()(GLIBCXX_3.4) [ISOCXX]

basic_ifstream<char, char_traits<char> >::basic_ifstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::basic_ifstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<char, char_traits<char> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ifstream<char, std::char_traits<char> > specified in Table 16-211, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-211 libstdcxx - Class basic_ifstream<char, char_traits<char> > Data Interfaces

typeinfo for basic_ifstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ifstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ifstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ifstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.79 Class basic_ifstream<wchar_t, char_traits<wchar_t> >

16.1.79.1 Class data for basic_ifstream<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_ifstream<wchar_t, std::char_traits<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ifstream<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-212

Table 16-212 VTT for basic_ifstream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt14basic_ifstreamIwSt11char_traitsIwEE
Number of Entries	4

16.1.79.2 Interfaces for Class basic_ifstream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ifstream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-213, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-213 libstdcxx - Class basic_ifstream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_ifstream<wchar_t, char_traits<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::is_open() const(GLIBCXX_3.4.5) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::is_open()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::basic_ifstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::basic_ifstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]
basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ifstream<wchar_t, std::char_traits<wchar_t> > specified in Table

16-214, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-214 libstdcxx - Class basic_ifstream<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_ifstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo for basic_streambuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ifstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_streambuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ifstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ifstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_streambuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.80 Class basic_ofstream<char, char_traits<char> >

16.1.80.1 Class data for basic_ofstream<char, char_traits<char> >

The virtual table for the std::basic_ofstream<char, std::char_traits<char> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ofstream<char, std::char_traits<char> > class is described by Table 16-215

Table 16-215 VTT for basic_ofstream<char, char_traits<char> >

VTT Name	_ZTTSt14basic_ofstreamIcSt11char_traitsIcEE
Number of Entries	4

16.1.80.2 Interfaces for Class basic_ofstream<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ofstream<char, std::char_traits<char> > specified in Table 16-216, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-216 libstdcxx - Class basic_ofstream<char, char_traits<char> > Function Interfaces

basic_ofstream<char, char_traits<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::is_open() const(GLIBCXX_3.4.5) [ISOCXX]
basic_ofstream<char, char_traits<char> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::is_open()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::basic_ofstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::basic_ofstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<char, char_traits<char> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ofstream<char, std::char_traits<char> > specified in Table 16-217, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-217 libstdcxx - Class basic_ofstream<char, char_traits<char> > Data Interfaces

typeinfo for basic_ofstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ofstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ofstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ofstream<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.81 Class basic_ofstream<wchar_t, char_traits<wchar_t> >

16.1.81.1 Class data for basic_ofstream<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_ofstream<wchar_t, std::char_traits<wchar_t> > class is described in the relevant architecture specific part of this specification.

The VTT for the std::basic_ofstream<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-218

Table 16-218 VTT for basic_ofstream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt14basic_ofstreamIwSt11char_traitsIwEE
Number of Entries	4

16.1.81.2 Interfaces for Class basic_ofstream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ofstream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-219, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-219 libstdcxx - Class basic_ofstream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_ofstream<wchar_t, char_traits<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::is_open() const(GLIBCXX_3.4.5) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::is_open()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::basic_ofstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::basic_ofstream(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::basic_ofstream()(GLIBCXX_3.4) [ISOCXX]

basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]
basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ofstream<wchar_t, std::char_traits<wchar_t> > specified in Table 16-220, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-220 libstdcxx - Class basic_ofstream<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_ofstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ofstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
VTT for basic_ofstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ofstream<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.82 Class basic_streambuf<char, char_traits<char> >

16.1.82.1 Class data for basic_streambuf<char, char_traits<char> >

The virtual table for the std::basic_streambuf<char, std::char_traits<char> > class is described by Table 16-221

Table 16-221 Primary vtable for basic_streambuf<char, char_traits<char> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_streambuf<char, char_traits<char> >
vfunc[0]:	basic_streambuf<char, char_traits<char> >::~basic_streambuf()
vfunc[1]:	basic_streambuf<char, char_traits<char> >::~basic_streambuf()

vfunc[2]:	basic_streambuf<char, char_traits<char> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	basic_streambuf<char, char_traits<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<char, char_traits<char> >::sync()
vfunc[7]:	basic_streambuf<char, char_traits<char> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	basic_streambuf<char, char_traits<char> >::underflow()
vfunc[10]:	basic_streambuf<char, char_traits<char> >::uflow()
vfunc[11]:	basic_streambuf<char, char_traits<char> >::pbackfail(int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	basic_streambuf<char, char_traits<char> >::overflow(int)

The Run Time Type Information for the std::basic_streambuf<char, std::char_traits<char> > class is described by Table 16-222

Table 16-222 typeinfo for basic_streambuf<char, char_traits<char> >

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for basic_streambuf<char, char_traits<char> >

16.1.82.2 Interfaces for Class basic_streambuf<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_streambuf<char, std::char_traits<char> > specified in Table 16-223, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-223 libstdcxx - Class basic_streambuf<char, char_traits<char> > Function Interfaces

basic_streambuf<char, char_traits<char> >::gptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::eback() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::egptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::epptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pbase() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::getloc() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pubseekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::setg(char*, char*, char*)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::setp(char*, char*)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sync()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::gbump(int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pbump(int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sgetc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sputc(char)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::uflow()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sbumpc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::snextc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pubsync()(GLIBCXX_3.4) [ISOCXX]

basic_streambuf<char, char_traits<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sungetc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::in_avail()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::overflow(int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pubimbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pbackfail(int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::showmanyc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sputbackc(char)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::underflow()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::basic_streambuf(basic_streambuf<char, char_traits<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::basic_streambuf(basic_streambuf<char, char_traits<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::~basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::~basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::~basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::operator=(basic_streambuf<char, char_traits<char> > const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_streambuf<char, std::char_traits<char> > specified in Table 16-224, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-224 libstdcxx - Class basic_streambuf<char, char_traits<char> > Data Interfaces

typeinfo for basic_streambuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_streambuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_streambuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.83 Class basic_streambuf<wchar_t, char_traits<wchar_t> >

16.1.83.1 Class data for basic_streambuf<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_streambuf<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-225

Table 16-225 Primary vtable for basic_streambuf<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_streambuf<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::~basic_streambuf()
vfunc[1]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::~basic_streambuf()
vfunc[2]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::sync()

vfunc[7]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::underflow()
vfunc[10]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)

The Run Time Type Information for the std::basic_streambuf<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-226

Table 16-226 typeinfo for basic_streambuf<wchar_t, char_traits<wchar_t> >

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for basic_streambuf<wchar_t, char_traits<wchar_t> >

16.1.83.2 Interfaces for Class basic_streambuf<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_streambuf<wchar_t, std::char_traits<wchar_t> > specified in Table 16-227, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-227 libstdcxx - Class basic_streambuf<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_streambuf<wchar_t, char_traits<wchar_t> >::gptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::eback() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::egptr() const(GLIBCXX_3.4) [ISOCXX]

basic_streambuf<wchar_t, char_traits<wchar_t> >::epptr() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pbase() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::getloc() const(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pubseekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::setg(wchar_t*, wchar_t*, wchar_t*)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::setp(wchar_t*, wchar_t*)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::sync()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::gbump(int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pbump(int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::sgetc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::sputc(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::sbumpc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::snextc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pubsync()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::sungetc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::in_avail()(GLIBCXX_3.4) [ISOCXX]

basic_streambuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pubimbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::sputbackc(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::underflow()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::basic_streambuf(basic_streambuf<wchar_t, char_traits<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::basic_streambuf(basic_streambuf<wchar_t, char_traits<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::~basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::~basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::~basic_streambuf()(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<wchar_t, char_traits<wchar_t> >::operator=(basic_streambuf<wchar_t, char_traits<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]

16.1.84 Class basic_filebuf<char, char_traits<char> >

16.1.84.1 Class data for basic_filebuf<char, char_traits<char> >

The virtual table for the std::basic_filebuf<char, std::char_traits<char> > class is described by Table 16-228

Table 16-228 Primary vtable for basic_filebuf<char, char_traits<char> >

Base Offset	0

Virtual Base Offset	0
RTTI	typeinfo for basic_filebuf<char, char_traits<char> >
vfunc[0]:	basic_filebuf<char, char_traits<char> >::~basic_filebuf()
vfunc[1]:	basic_filebuf<char, char_traits<char> >::~basic_filebuf()
vfunc[2]:	basic_filebuf<char, char_traits<char> >::imbue(locale const&)
vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	basic_filebuf<char, char_traits<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_filebuf<char, char_traits<char> >::sync()
vfunc[7]:	basic_filebuf<char, char_traits<char> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	basic_filebuf<char, char_traits<char> >::underflow()
vfunc[10]:	basic_streambuf<char, char_traits<char> >::uflow()
vfunc[11]:	basic_filebuf<char, char_traits<char> >::pbackfail(int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	basic_filebuf<char, char_traits<char> >::overflow(int)

The Run Time Type Information for the std::basic_filebuf<char, std::char_traits<char> > class is described by Table 16-229

Table 16-229 typeinfo for basic_filebuf<char, char_traits<char> >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for basic_filebuf<char, char_traits<char> >

16.1.84.2 Interfaces for Class basic_filebuf<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_filebuf<char, std::char_traits<char> > specified in Table 16-230, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-230 libstdcxx - Class basic_filebuf<char, char_traits<char> > Function Interfaces

basic_filebuf<char, char_traits<char> >::is_open() const(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_create_pback()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_destroy_pback()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_terminate_output()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_destroy_internal_buffer()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_allocate_internal_buffer()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::sync()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::overflow(int)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::pbackfail(int)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::showmanyc()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::underflow()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::basic_filebuf()(GLIBCXX_3.4) [ISOCXX]

basic_filebuf<char, char_traits<char> >::~basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::~basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::~basic_filebuf()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_filebuf<char, std::char_traits<char> > specified in Table 16-231, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-231 libstdcxx - Class basic_filebuf<char, char_traits<char> > Data Interfaces

typeinfo for basic_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_filebuf<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.85 Class basic_filebuf<wchar_t, char_traits<wchar_t> >

16.1.85.1 Class data for basic_filebuf<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_filebuf<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-232

Table 16-232 Primary vtable for basic_filebuf<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_filebuf<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::~basic_filebuf()
vfunc[1]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::~basic_filebuf()
vfunc[2]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)

vfunc[3]:	See architecture specific part.
vfunc[4]:	See architecture specific part.
vfunc[5]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	See architecture specific part.
vfunc[9]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::underflow()
vfunc[10]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	See architecture specific part.
vfunc[13]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)

The Run Time Type Information for the std::basic_filebuf<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-233

Table 16-233 typeinfo for basic_filebuf<wchar_t, char_traits<wchar_t> >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for basic_filebuf<wchar_t, char_traits<wchar_t> >

16.1.85.2 Interfaces for Class basic_filebuf<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_filebuf<wchar_t, std::char_traits<wchar_t> > specified in Table 16-234, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-234 libstdcxx - Class basic_filebuf<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_filebuf<wchar_t, char_traits<wchar_t> >::is_open() const(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_create_pback()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_destroy_pback()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_terminate_output()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_destroy_internal_buffer()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_allocate_internal_buffer()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::open(char const*, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::sync()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::close()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::overflow(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::pbackfail(unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::showmanyc()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::underflow()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::~basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::~basic_filebuf()(GLIBCXX_3.4) [ISOCXX]

basic_filebuf<wchar_t, char_traits<wchar_t> >::~basic_filebuf()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::basic_istream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::basic_ostream()(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::basic_ostream()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_filebuf<wchar_t, std::char_traits<wchar_t> > specified in Table 16-235, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-235 libstdcxx - Class basic_filebuf<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_filebuf<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.86 Class ios_base

16.1.86.1 Class data for ios_base

The virtual table for the std::ios_base class is described by Table 16-236

Table 16-236 Primary vtable for ios_base

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ios_base
vfunc[0]:	ios_base::~ios_base()
vfunc[1]:	ios_base::~ios_base()

The Run Time Type Information for the std::ios_base class is described by Table 16-237

Table 16-237 typeinfo for ios_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for ios_base

16.1.86.2 Interfaces for Class ios_base

An LSB conforming implementation shall provide the generic methods for Class std::ios_base specified in Table 16-238, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-238 libstdcxx - Class ios_base Function Interfaces

ios_base::_M_grow_words(int, bool)(GLIBCXX_3.4) [ISOCXX]
ios_base::sync_with_stdio(bool)(GLIBCXX_3.4) [ISOCXX]
ios_base::_M_call_callbacks(ios_base::event)(GLIBCXX_3.4.6) [ISOCXX]
ios_base::register_callback(void (*)(ios_base::event, ios_base&, int), int)(GLIBCXX_3.4) [ISOCXX]
ios_base::_M_dispose_callbacks()(GLIBCXX_3.4.6) [ISOCXX]
ios_base::Init::Init()(GLIBCXX_3.4) [ISOCXX]
ios_base::Init::Init()(GLIBCXX_3.4) [ISOCXX]
ios_base::Init::~Init()(GLIBCXX_3.4) [ISOCXX]
ios_base::Init::~Init()(GLIBCXX_3.4) [ISOCXX]
ios_base::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
ios_base::xalloc()(GLIBCXX_3.4) [ISOCXX]
ios_base::_M_init()(GLIBCXX_3.4) [ISOCXX]
ios_base::failure::failure(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
ios_base::failure::failure(basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
ios_base::failure::~failure()(GLIBCXX_3.4) [ISOCXX]
ios_base::failure::~failure()(GLIBCXX_3.4) [ISOCXX]
ios_base::failure::~failure()(GLIBCXX_3.4) [ISOCXX]
ios_base::ios_base()(GLIBCXX_3.4) [ISOCXX]
ios_base::ios_base()(GLIBCXX_3.4) [ISOCXX]
ios_base::~ios_base()(GLIBCXX_3.4) [ISOCXX]
ios_base::~ios_base()(GLIBCXX_3.4) [ISOCXX]
ios_base::~ios_base()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::ios_base specified in Table 16-239, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-239 libstdcxx - Class ios_base Data Interfaces

ios_base::floatfield(GLIBCXX_3.4) [ISOCXX]
ios_base::scientific(GLIBCXX_3.4) [ISOCXX]
ios_base::adjustfield(GLIBCXX_3.4) [ISOCXX]
ios_base::in(GLIBCXX_3.4) [ISOCXX]
ios_base::app(GLIBCXX_3.4) [ISOCXX]
ios_base::ate(GLIBCXX_3.4) [ISOCXX]
ios_base::beg(GLIBCXX_3.4) [ISOCXX]
ios_base::cur(GLIBCXX_3.4) [ISOCXX]
ios_base::dec(GLIBCXX_3.4) [ISOCXX]
ios_base::end(GLIBCXX_3.4) [ISOCXX]
ios_base::hex(GLIBCXX_3.4) [ISOCXX]
ios_base::oct(GLIBCXX_3.4) [ISOCXX]
ios_base::out(GLIBCXX_3.4) [ISOCXX]
ios_base::left(GLIBCXX_3.4) [ISOCXX]
ios_base::fixed(GLIBCXX_3.4) [ISOCXX]
ios_base::right(GLIBCXX_3.4) [ISOCXX]
ios_base::trunc(GLIBCXX_3.4) [ISOCXX]
ios_base::badbit(GLIBCXX_3.4) [ISOCXX]
ios_base::binary(GLIBCXX_3.4) [ISOCXX]
ios_base::eofbit(GLIBCXX_3.4) [ISOCXX]
ios_base::skipws(GLIBCXX_3.4) [ISOCXX]
ios_base::failbit(GLIBCXX_3.4) [ISOCXX]
ios_base::goodbit(GLIBCXX_3.4) [ISOCXX]
ios_base::showpos(GLIBCXX_3.4) [ISOCXX]
ios_base::unitbuf(GLIBCXX_3.4) [ISOCXX]
ios_base::internal(GLIBCXX_3.4) [ISOCXX]
ios_base::showbase(GLIBCXX_3.4) [ISOCXX]
ios_base::basefield(GLIBCXX_3.4) [ISOCXX]
ios_base::boolalpha(GLIBCXX_3.4) [ISOCXX]

ios_base::showpoint(GLIBCXX_3.4) [ISOCXX]
ios_base::uppercase(GLIBCXX_3.4) [ISOCXX]
typeinfo for ios_base(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ios_base(GLIBCXX_3.4) [CXXABI-1.86]
vtable for ios_base(GLIBCXX_3.4) [CXXABI-1.86]

16.1.87 Class basic_ios<char, char_traits<char> >

16.1.87.1 Class data for basic_ios<char, char_traits<char> >

The virtual table for the std::basic_ios<char, std::char_traits<char> > class is described by Table 16-240

Table 16-240 Primary vtable for basic_ios<char, char_traits<char> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_ios<char, char_traits<char> >
vfunc[0]:	basic_ios<char, char_traits<char> >::~basic_ios()
vfunc[1]:	basic_ios<char, char_traits<char> >::~basic_ios()

16.1.87.2 Interfaces for Class basic_ios<char, char_traits<char> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ios<char, std::char_traits<char> > specified in Table 16-241, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-241 libstdcxx - Class basic_ios<char, char_traits<char> > Function Interfaces

basic_ios<char, char_traits<char> >::exceptions() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::bad() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::eof() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::tie() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::fail() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::fill() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::good() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]

basic_ios<char, char_traits<char> >::widen(char) const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::narrow(char, char) const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::rdstate() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::operator void*() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::operator!() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::exceptions(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::_M_setstate(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::tie(basic_ostream<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::fill(char)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::init(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::clear(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::rdbuf(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::copyfmt(basic_ios<char, char_traits<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::setstate(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::basic_ios(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::basic_ios(basic_streambuf<char, char_traits<char> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::~basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::~basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<char, char_traits<char> >::~basic_ios()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ios<char, std::char_traits<char> > specified in Table 16-242, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-242 libstdcxx - Class basic_ios<char, char_traits<char> > Data Interfaces

typeinfo for basic_ios<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ios<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for basic_ios<char, char_traits<char> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.88 Class basic_ios<wchar_t, char_traits<wchar_t> >

16.1.88.1 Class data for basic_ios<wchar_t, char_traits<wchar_t> >

The virtual table for the std::basic_ios<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-243

Table 16-243 Primary vtable for basic_ios<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for basic_ios<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_ios<wchar_t, char_traits<wchar_t> >::~basic_ios()
vfunc[1]:	basic_ios<wchar_t, char_traits<wchar_t> >::~basic_ios()

The Run Time Type Information for the std::basic_ios<wchar_t, std::char_traits<wchar_t> > class is described by Table 16-244

Table 16-244 typeinfo for basic_ios<wchar_t, char_traits<wchar_t> >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info	
Name	typeinfo name for basic_ios<wchar_t, char_traits<wchar_t> >	
flags:	8	
basetype:	typeinfo for ios_base	1026

16.1.88.2 Interfaces for Class basic_ios<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the generic methods for Class std::basic_ios<wchar_t, std::char_traits<wchar_t> > specified in Table 16-245, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-245 libstdcxx - Class basic_ios<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_ios<wchar_t, char_traits<wchar_t> >::exceptions() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::bad() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::eof() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::tie() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::fail() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::fill() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::good() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::rdbuf() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::widen(char) const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::narrow(wchar_t, char) const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::rdstate() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::operator void*() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::operator!() const(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::exceptions(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::_M_setstate(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::tie(basic_ostream<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]

basic_ios<wchar_t, char_traits<wchar_t> >::fill(wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::init(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::clear(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::imbue(locale const&)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::rdbuf(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::copyfmt(basic_ios<wchar_t, char_traits<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::setstate(_Ios_Iostate)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::basic_ios(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::basic_ios(basic_streambuf<wchar_t, char_traits<wchar_t> >*)(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::~basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::~basic_ios()(GLIBCXX_3.4) [ISOCXX]
basic_ios<wchar_t, char_traits<wchar_t> >::~basic_ios()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::basic_ios<wchar_t, std::char_traits<wchar_t> > specified in Table 16-246, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-246 libstdcxx - Class basic_ios<wchar_t, char_traits<wchar_t> > Data Interfaces

typeinfo for basic_ios<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for basic_ios<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

vtable for basic_ios<wchar_t, char_traits<wchar_t> >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.89 Class ios_base::failure

16.1.89.1 Class data for ios_base::failure

The virtual table for the std::ios_base::failure class is described by Table 16-247

Table 16-247 Primary vtable for ios_base::failure

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for ios_base::failure
vfunc[0]:	ios_base::failure::~failure()
vfunc[1]:	ios_base::failure::~failure()
vfunc[2]:	ios_base::failure::what() const

The Run Time Type Information for the std::ios_base::failure class is described by Table 16-248

Table 16-248 typeinfo for ios_base::failure

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for ios_base::failure

16.1.89.2 Interfaces for Class ios_base::failure

An LSB conforming implementation shall provide the generic methods for Class std::ios_base::failure specified in Table 16-249, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-249 libstdcxx - Class ios_base::failure Function Interfaces

ios_base::failure::what() const(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::ios_base::failure specified in Table 16-250, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-250 libstdcxx - Class ios_base::failure Data Interfaces

typeinfo for ios_base::failure(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for ios_base::failure(GLIBCXX_3.4) [CXXABI-1.86]
vtable for ios_base::failure(GLIBCXX_3.4) [CXXABI-1.86]

16.1.90 Class __timepunct<char>

16.1.90.1 Class data for __timepunct<char>

The virtual table for the std::__timepunct<char> class is described by Table 16-251

Table 16-251 Primary vtable for __timepunct<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __timepunct<char>
vfunc[0]:	__timepunct<char>::~__timepunct()
vfunc[1]:	__timepunct<char>::~__timepunct()

The Run Time Type Information for the std::__timepunct<char> class is described by Table 16-252

Table 16-252 typeinfo for __timepunct<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __timepunct<char>

16.1.90.2 Interfaces for Class __timepunct<char>

An LSB conforming implementation shall provide the generic methods for Class std::__timepunct<char> specified in Table 16-253, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-253 libstdcxx - Class __timepunct<char> Function Interfaces

__timepunct<char>::_M_am_pm_format(char const*) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_date_formats(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_time_formats(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_days_abbreviated(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_date_time_formats(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_months_abbreviated(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_days(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_am_pm(char const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_months(char const**) const(GLIBCXX_3.4) [ISOCXX]

__timepunct<wchar_t>::_M_am_pm_format(wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::_M_initialize_timepunct(__locale_struct*)(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::~__timepunct()(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::~__timepunct()(GLIBCXX_3.4) [ISOCXX]
__timepunct<char>::~__timepunct()(GLIBCXX_3.4) [ISOCXX]
bool has_facet<__timepunct<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::__timepunct<char> specified in Table 16-254, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-254 libstdcxx - Class __timepunct<char> Data Interfaces

guard variable for __timepunct<char>::id(GLIBCXX_3.4) [CXXABI-1.86]
__timepunct<char>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for __timepunct<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __timepunct<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __timepunct<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.91 Class __timepunct<wchar_t>

16.1.91.1 Class data for __timepunct<wchar_t>

The virtual table for the std::__timepunct<wchar_t> class is described by Table 16-255

Table 16-255 Primary vtable for __timepunct<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __timepunct<wchar_t>
vfunc[0]:	__timepunct<wchar_t>::~__timepunct()
vfunc[1]:	__timepunct<wchar_t>::~__timepunct()

The Run Time Type Information for the std::__timepunct<wchar_t> class is described by Table 16-256

Table 16-256 typeinfo for __timepunct<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info

Name	typeinfo name for __timepunct<wchar_t>

16.1.91.2 Interfaces for Class __timepunct<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::__timepunct<wchar_t> specified in Table 16-257, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-257 libstdcxx - Class __timepunct<wchar_t> Function Interfaces

__timepunct<wchar_t>::_M_date_formats(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_time_formats(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_days_abbreviated(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_date_time_formats(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_months_abbreviated(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_days(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_am_pm(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_months(wchar_t const**) const(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::_M_initialize_timepunct(__locale_struct*)(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::~__timepunct()(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::~__timepunct()(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::~__timepunct()(GLIBCXX_3.4) [ISOCXX]
bool has_facet<__timepunct<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::__timepunct<wchar_t> specified in Table 16-258, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-258 libstdcxx - Class __timepunct<wchar_t> Data Interfaces

guard variable for __timepunct<wchar_t>::id(GLIBCXX_3.4) [CXXABI-1.86]
__timepunct<wchar_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for __timepunct<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for __timepunct<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __timepunct<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.92 Class messages_base

16.1.92.1 Class data for messages_base

The Run Time Type Information for the std::messages_base class is described by Table 16-259

Table 16-259 typeinfo for messages_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for messages_base

16.1.92.2 Interfaces for Class messages_base

No external methods are defined for libstdcxx - Class std::messages_base in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::messages_base specified in Table 16-260, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-260 libstdcxx - Class messages_base Data Interfaces

typeinfo for messages_base(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for messages_base(GLIBCXX_3.4) [CXXABI-1.86]

16.1.93 Class messages<char>

16.1.93.1 Class data for messages<char>

The virtual table for the std::messages<char> class is described by Table 16-261

Table 16-261 Primary vtable for messages<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for messages<char>
vfunc[0]:	messages<char>::~messages()
vfunc[1]:	messages<char>::~messages()
vfunc[2]:	messages<char>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const

vfunc[3]:	messages<char>::do_get(int, int, int, basic_string<char, char_traits<char>, allocator<char> > const&) const
vfunc[4]:	messages<char>::do_close(int) const

16.1.93.2 Interfaces for Class messages<char>

An LSB conforming implementation shall provide the generic methods for Class std::messages<char> specified in Table 16-262, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-262 libstdcxx - Class messages<char> Function Interfaces

messages<char>::_M_convert_to_char(basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::_M_convert_from_char(char*) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::get(int, int, int, basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&, char const*) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::close(int) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::do_get(int, int, int, basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::do_close(int) const(GLIBCXX_3.4) [ISOCXX]
messages<char>::~messages()(GLIBCXX_3.4) [ISOCXX]
messages<char>::~messages()(GLIBCXX_3.4) [ISOCXX]
messages<char>::~messages()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::messages<char> specified in Table 16-263, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-263 libstdcxx - Class messages<char> Data Interfaces

guard variable for messages<char>::id(GLIBCXX_3.4) [CXXABI-1.86]
messages<char>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for messages<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for messages<char>(GLIBCXX_3.4) [CXXABI-1.86]

vtable for messages<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.94 Class messages<wchar_t>

16.1.94.1 Class data for messages<wchar_t>

The virtual table for the std::messages<wchar_t> class is described by Table 16-264

Table 16-264 Primary vtable for messages<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for messages<wchar_t>
vfunc[0]:	messages<wchar_t>::~messages()
vfunc[1]:	messages<wchar_t>::~messages()
vfunc[2]:	messages<wchar_t>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const
vfunc[3]:	messages<wchar_t>::do_get(int, int, int, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const
vfunc[4]:	messages<wchar_t>::do_close(int) const

16.1.94.2 Interfaces for Class messages<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::messages<wchar_t> specified in Table 16-265, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-265 libstdcxx - Class messages<wchar_t> Function Interfaces

messages<wchar_t>::_M_convert_to_char(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::_M_convert_from_char(char*) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::get(int, int, int, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const(GLIBCXX_3.4) [ISOCXX]

messages<wchar_t>::open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&, char const*) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::close(int) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::do_get(int, int, int, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::do_close(int) const(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::~messages()(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::~messages()(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::~messages()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::messages<wchar_t> specified in Table 16-266, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-266 libstdcxx - Class messages<wchar_t> Data Interfaces

guard variable for messages<wchar_t>::id(GLIBCXX_3.4) [CXXABI-1.86]
messages<wchar_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for messages<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for messages<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for messages<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.95 Class messages_byname<char>

16.1.95.1 Class data for messages_byname<char>

The virtual table for the std::messages_byname<char> class is described by Table 16-267

Table 16-267 Primary vtable for messages_byname<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for messages_byname<char>
vfunc[0]:	messages_byname<char>::~messages_byname()
vfunc[1]:	messages_byname<char>::~messages_byname()

vfunc[2]:	messages<char>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const
vfunc[3]:	messages<char>::do_get(int, int, int, basic_string<char, char_traits<char>, allocator<char> > const&) const
vfunc[4]:	messages<char>::do_close(int) const

The Run Time Type Information for the std::messages_byname<char> class is described by Table 16-268

Table 16-268 typeinfo for messages_byname<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for messages_byname<char>

16.1.95.2 Interfaces for Class messages_byname<char>

An LSB conforming implementation shall provide the generic methods for Class std::messages_byname<char> specified in Table 16-269, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-269 libstdcxx - Class messages_byname<char> Function Interfaces

messages_byname<char>::~messages_byname()(GLIBCXX_3.4) [ISOCXX]
messages_byname<char>::~messages_byname()(GLIBCXX_3.4) [ISOCXX]
messages_byname<char>::~messages_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::messages_byname<char> specified in Table 16-270, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-270 libstdcxx - Class messages_byname<char> Data Interfaces

typeinfo for messages_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for messages_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for messages_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.96 Class messages_byname<wchar_t>

16.1.96.1 Class data for messages_byname<wchar_t>

The virtual table for the std::messages_byname<wchar_t> class is described by Table 16-271

Table 16-271 Primary vtable for messages_byname<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for messages_byname<wchar_t>
vfunc[0]:	messages_byname<wchar_t>::~messages_byname()
vfunc[1]:	messages_byname<wchar_t>::~messages_byname()
vfunc[2]:	messages<wchar_t>::do_open(basic_string<char, char_traits<char>, allocator<char> > const&, locale const&) const
vfunc[3]:	messages<wchar_t>::do_get(int, int, int, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const
vfunc[4]:	messages<wchar_t>::do_close(int) const

The Run Time Type Information for the std::messages_byname<wchar_t> class is described by Table 16-272

Table 16-272 typeinfo for messages_byname<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for messages_byname<wchar_t>

16.1.96.2 Interfaces for Class messages_byname<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::messages_byname<wchar_t> specified in Table 16-273, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-273 libstdcxx - Class messages_byname<wchar_t> Function Interfaces

messages_byname<wchar_t>::~messages_byname()(GLIBCXX_3.4) [ISOCXX]
messages_byname<wchar_t>::~messages_byname()(GLIBCXX_3.4) [ISOCXX]
messages_byname<wchar_t>::~messages_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::messages_byname<wchar_t> specified in Table 16-274, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-274 libstdcxx - Class messages_byname<wchar_t> Data Interfaces

typeinfo for messages_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for messages_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for messages_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.97 Class numpunct<char>

16.1.97.1 Class data for numpunct<char>

The virtual table for the std::numpunct<char> class is described by Table 16-275

Table 16-275 Primary vtable for numpunct<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for numpunct<char>
vfunc[0]:	numpunct<char>::~numpunct()
vfunc[1]:	numpunct<char>::~numpunct()
vfunc[2]:	numpunct<char>::do_decimal_point() const
vfunc[3]:	numpunct<char>::do_thousands_sep() const
vfunc[4]:	numpunct<char>::do_grouping() const
vfunc[5]:	numpunct<char>::do_truename() const
vfunc[6]:	numpunct<char>::do_falsename() const

The Run Time Type Information for the std::numpunct<char> class is described by Table 16-276

Table 16-276 typeinfo for numpunct<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for numpunct<char>

16.1.97.2 Interfaces for Class numpunct<char>

An LSB conforming implementation shall provide the generic methods for Class std::numpunct<char> specified in Table 16-277, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-277 libstdcxx - Class numpunct<char> Function Interfaces

numpunct<char>::do_grouping() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::do_truename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::do_falsename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::decimal_point() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::do_decimal_point() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::do_thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::grouping() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::truename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::falsename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::_M_initialize_numpunct(__locale_struct*)(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::~numpunct()(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::~numpunct()(GLIBCXX_3.4) [ISOCXX]
numpunct<char>::~numpunct()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::numpunct<char> specified in Table 16-278, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-278 libstdcxx - Class numpunct<char> Data Interfaces

guard variable for numpunct<char>::id(GLIBCXX_3.4) [CXXABI-1.86]
numpunct<char>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for numpunct<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for numpunct<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for numpunct<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.98 Class numpunct<wchar_t>

16.1.98.1 Class data for numpunct<wchar_t>

The virtual table for the std::numpunct<wchar_t> class is described by Table 16-279

Table 16-279 Primary vtable for numpunct<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for numpunct<wchar_t>

vfunc[0]:	numpunct<wchar_t>::~numpunct()
vfunc[1]:	numpunct<wchar_t>::~numpunct()
vfunc[2]:	numpunct<wchar_t>::do_decimal_point() const
vfunc[3]:	numpunct<wchar_t>::do_thousands_sep() const
vfunc[4]:	numpunct<wchar_t>::do_grouping() const
vfunc[5]:	numpunct<wchar_t>::do_truename() const
vfunc[6]:	numpunct<wchar_t>::do_falsename() const

The Run Time Type Information for the std::numpunct<wchar_t> class is described by Table 16-280

Table 16-280 typeinfo for numpunct<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for numpunct<wchar_t>

16.1.98.2 Interfaces for Class numpunct<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::numpunct<wchar_t> specified in Table 16-281, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-281 libstdcxx - Class numpunct<wchar_t> Function Interfaces

numpunct<wchar_t>::do_grouping() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::do_truename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::do_falsename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::decimal_point() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::do_decimal_point() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::do_thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::grouping() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::truename() const(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::falsename() const(GLIBCXX_3.4) [ISOCXX]

numpunct<wchar_t>::_M_initialize_numpunct(__locale_struct*)(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::~numpunct()(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::~numpunct()(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t>::~numpunct()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::numpunct<wchar_t> specified in Table 16-282, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-282 libstdcxx - Class numpunct<wchar_t> Data Interfaces

guard variable for numpunct<wchar_t>::id(GLIBCXX_3.4) [CXXABI-1.86]
numpunct<wchar_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for numpunct<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for numpunct<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for numpunct<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.99 Class numpunct_byname<char>

16.1.99.1 Class data for numpunct_byname<char>

The virtual table for the std::numpunct_byname<char> class is described by Table 16-283

Table 16-283 Primary vtable for numpunct_byname<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for numpunct_byname<char>
vfunc[0]:	numpunct_byname<char>::~numpunct_byname()
vfunc[1]:	numpunct_byname<char>::~numpunct_byname()
vfunc[2]:	numpunct<char>::do_decimal_point() const
vfunc[3]:	numpunct<char>::do_thousands_sep() const
vfunc[4]:	numpunct<char>::do_grouping() const
vfunc[5]:	numpunct<char>::do_truename() const

vfunc[6]:	numpunct<char>::do_falsename() const

The Run Time Type Information for the std::numpunct_byname<char> class is described by Table 16-284

Table 16-284 typeinfo for numpunct_byname<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for numpunct_byname<char>

16.1.99.2 Interfaces for Class numpunct_byname<char>

An LSB conforming implementation shall provide the generic methods for Class std::numpunct_byname<char> specified in Table 16-285, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-285 libstdcxx - Class numpunct_byname<char> Function Interfaces

numpunct_byname<char>::~numpunct_byname()(GLIBCXX_3.4) [ISOCXX]
numpunct_byname<char>::~numpunct_byname()(GLIBCXX_3.4) [ISOCXX]
numpunct_byname<char>::~numpunct_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::numpunct_byname<char> specified in Table 16-286, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-286 libstdcxx - Class numpunct_byname<char> Data Interfaces

typeinfo for numpunct_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for numpunct_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for numpunct_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.100 Class numpunct_byname<wchar_t>

16.1.100.1 Class data for numpunct_byname<wchar_t>

The virtual table for the std::numpunct_byname<wchar_t> class is described by Table 16-287

Table 16-287 Primary vtable for numpunct_byname<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for numpunct_byname<wchar_t>

vfunc[0]:	numpunct_byname<wchar_t>::~numpunct_byname()
vfunc[1]:	numpunct_byname<wchar_t>::~numpunct_byname()
vfunc[2]:	numpunct<wchar_t>::do_decimal_point() const
vfunc[3]:	numpunct<wchar_t>::do_thousands_sep() const
vfunc[4]:	numpunct<wchar_t>::do_grouping() const
vfunc[5]:	numpunct<wchar_t>::do_truename() const
vfunc[6]:	numpunct<wchar_t>::do_falsename() const

The Run Time Type Information for the std::numpunct_byname<wchar_t> class is described by Table 16-288

Table 16-288 typeinfo for numpunct_byname<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for numpunct_byname<wchar_t>

16.1.100.2 Interfaces for Class numpunct_byname<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::numpunct_byname<wchar_t> specified in Table 16-289, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-289 libstdcxx - Class numpunct_byname<wchar_t> Function Interfaces

numpunct_byname<wchar_t>::~numpunct_byname()(GLIBCXX_3.4) [ISOCXX]
numpunct_byname<wchar_t>::~numpunct_byname()(GLIBCXX_3.4) [ISOCXX]
numpunct_byname<wchar_t>::~numpunct_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::numpunct_byname<wchar_t> specified in Table 16-290, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-290 libstdcxx - Class numpunct_byname<wchar_t> Data Interfaces

typeinfo for numpunct_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for numpunct_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for numpunct_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.101 Class __codecvt_abstract_base<char, char, __mbstate_t>

16.1.101.1 Class data for __codecvt_abstract_base<char, char, __mbstate_t>

The virtual table for the std::__codecvt_abstract_base<char, char, __mbstate_t> class is described by Table 16-291

Table 16-291 Primary vtable for __codecvt_abstract_base<char, char, __mbstate_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __codecvt_abstract_base<char, char, __mbstate_t>
vfunc[0]:	NULL or __codecvt_abstract_base<char, char, __mbstate_t>::~__codecvt_abstract_base()
vfunc[1]:	NULL or __codecvt_abstract_base<char, char, __mbstate_t>::~__codecvt_abstract_base()
vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual

16.1.101.2 Interfaces for Class __codecvt_abstract_base<char, char, __mbstate_t>

No external methods are defined for libstdcxx - Class std::__codecvt_abstract_base<char, char, __mbstate_t> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::__codecvt_abstract_base<char, char, __mbstate_t> specified in Table 16-292, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-292 libstdcxx - Class __codecvt_abstract_base<char, char, __mbstate_t> Data Interfaces

typeinfo for __codecvt_abstract_base<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __codecvt_abstract_base<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __codecvt_abstract_base<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.102 Class __codecvt_abstract_base<wchar_t, char, __mbstate_t>

16.1.102.1 Class data for __codecvt_abstract_base<wchar_t, char, __mbstate_t>

The virtual table for the std::__codecvt_abstract_base<wchar_t, char, __mbstate_t> class is described by Table 16-293

Table 16-293 Primary vtable for __codecvt_abstract_base<wchar_t, char, __mbstate_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for __codecvt_abstract_base<wchar_t, char, __mbstate_t>
vfunc[0]:	NULL or __codecvt_abstract_base<wchar_t, char, __mbstate_t>::~__codecvt_abstract_base()
vfunc[1]:	NULL or __codecvt_abstract_base<wchar_t, char, __mbstate_t>::~__codecvt_abstract_base()

vfunc[2]:	__cxa_pure_virtual
vfunc[3]:	__cxa_pure_virtual
vfunc[4]:	__cxa_pure_virtual
vfunc[5]:	__cxa_pure_virtual
vfunc[6]:	__cxa_pure_virtual
vfunc[7]:	__cxa_pure_virtual
vfunc[8]:	__cxa_pure_virtual

16.1.102.2 Interfaces for Class __codecvt_abstract_base<wchar_t, char, __mbstate_t>

No external methods are defined for libstdcxx - Class std::__codecvt_abstract_base<wchar_t, char, __mbstate_t> in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::__codecvt_abstract_base<wchar_t, char, __mbstate_t> specified in Table 16-294, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-294 libstdcxx - Class __codecvt_abstract_base<wchar_t, char, __mbstate_t> Data Interfaces

typeinfo for __codecvt_abstract_base<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for __codecvt_abstract_base<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for __codecvt_abstract_base<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.103 Class codecvt_base

16.1.103.1 Class data for codecvt_base

The Run Time Type Information for the std::codecvt_base class is described by Table 16-295

Table 16-295 typeinfo for codecvt_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for codecvt_base

16.1.103.2 Interfaces for Class codecvt_base

No external methods are defined for libstdcxx - Class std::codecvt_base in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::codecvt_base specified in Table 16-296, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-296 libstdcxx - Class codecvt_base Data Interfaces

typeinfo for codecvt_base(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for codecvt_base(GLIBCXX_3.4) [CXXABI-1.86]

16.1.104 Class codecvt<char, char, __mbstate_t>

16.1.104.1 Class data for codecvt<char, char, __mbstate_t>

The virtual table for the std::codecvt<char, char, __mbstate_t> class is described by Table 16-297

Table 16-297 Primary vtable for codecvt<char, char, __mbstate_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for codecvt<char, char, __mbstate_t>
vfunc[0]:	codecvt<char, char, __mbstate_t>::~codecvt()
vfunc[1]:	codecvt<char, char, __mbstate_t>::~codecvt()
vfunc[2]:	codecvt<char, char, __mbstate_t>::do_out(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const
vfunc[3]:	codecvt<char, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const
vfunc[4]:	codecvt<char, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const
vfunc[5]:	codecvt<char, char, __mbstate_t>::do_encoding() const
vfunc[6]:	codecvt<char, char, __mbstate_t>::do_always_noconv() const
vfunc[7]:	See architecture specific part.
vfunc[8]:	codecvt<char, char, __mbstate_t>::do_max_length() const

The Run Time Type Information for the std::codecvt<char, char, __mbstate_t> class is described by Table 16-298

Table 16-298 typeinfo for codecvt<char, char, __mbstate_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for codecvt<char, char, __mbstate_t>

16.1.104.2 Interfaces for Class codecvt<char, char, __mbstate_t>

An LSB conforming implementation shall provide the generic methods for Class std::codecvt<char, char, __mbstate_t> specified in Table 16-299, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-299 libstdcxx - Class codecvt<char, char, __mbstate_t> Function Interfaces

codecvt<char, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::do_encoding() const(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::do_max_length() const(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::do_always_noconv() const(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::do_out(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::~codecvt()(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::~codecvt()(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t>::~codecvt()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::codecvt<char, char, __mbstate_t> specified in Table 16-300, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-300 libstdcxx - Class codecvt<char, char, __mbstate_t> Data Interfaces

codecvt<char, char, __mbstate_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for codecvt<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for codecvt<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

vtable for codecvt<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.105 Class codecvt<wchar_t, char, __mbstate_t>

16.1.105.1 Class data for codecvt<wchar_t, char, __mbstate_t>

The virtual table for the std::codecvt<wchar_t, char, __mbstate_t> class is described by Table 16-301

Table 16-301 Primary vtable for codecvt<wchar_t, char, __mbstate_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for codecvt<wchar_t, char, __mbstate_t>
vfunc[0]:	codecvt<wchar_t, char, __mbstate_t>::~codecvt()
vfunc[1]:	codecvt<wchar_t, char, __mbstate_t>::~codecvt()
vfunc[2]:	codecvt<wchar_t, char, __mbstate_t>::do_out(__mbstate_t&, wchar_t const*, wchar_t const*, wchar_t const*&, char*, char*, char*&) const
vfunc[3]:	codecvt<wchar_t, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const
vfunc[4]:	codecvt<wchar_t, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, wchar_t*, wchar_t*, wchar_t*&) const
vfunc[5]:	codecvt<wchar_t, char, __mbstate_t>::do_encoding() const
vfunc[6]:	codecvt<wchar_t, char, __mbstate_t>::do_always_noconv() const
vfunc[7]:	See architecture specific part.
vfunc[8]:	codecvt<wchar_t, char, __mbstate_t>::do_max_length() const

The Run Time Type Information for the std::codecvt<wchar_t, char, __mbstate_t> class is described by Table 16-302

Table 16-302 typeinfo for codecvt<wchar_t, char, __mbstate_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for codecvt<wchar_t, char, __mbstate_t>

16.1.105.2 Interfaces for Class codecvt<wchar_t, char, __mbstate_t>

An LSB conforming implementation shall provide the generic methods for Class std::codecvt<wchar_t, char, __mbstate_t> specified in Table 16-303, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-303 libstdcxx - Class codecvt<wchar_t, char, __mbstate_t> Function Interfaces

codecvt<wchar_t, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::do_encoding() const(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::do_max_length() const(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::do_always_noconv() const(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, wchar_t*, wchar_t*, wchar_t*&) const(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::do_out(__mbstate_t&, wchar_t const*, wchar_t const*, wchar_t const*&, char*, char*, char*&) const(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::~codecvt()(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::~codecvt()(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t>::~codecvt()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::codecvt<wchar_t, char, __mbstate_t> specified in Table 16-304, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-304 libstdcxx - Class codecvt<wchar_t, char, __mbstate_t> Data Interfaces

codecvt<wchar_t, char, __mbstate_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for codecvt<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for codecvt<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for codecvt<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.106 Class codecvt_byname<char, char, __mbstate_t>

16.1.106.1 Class data for codecvt_byname<char, char, __mbstate_t>

The virtual table for the std::codecvt_byname<char, char, __mbstate_t> class is described by Table 16-305

Table 16-305 Primary vtable for codecvt_byname<char, char, __mbstate_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for codecvt_byname<char, char, __mbstate_t>
vfunc[0]:	codecvt_byname<char, char, __mbstate_t>::~codecvt_byname()
vfunc[1]:	codecvt_byname<char, char, __mbstate_t>::~codecvt_byname()
vfunc[2]:	codecvt<char, char, __mbstate_t>::do_out(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const
vfunc[3]:	codecvt<char, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const
vfunc[4]:	codecvt<char, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const
vfunc[5]:	codecvt<char, char, __mbstate_t>::do_encoding() const
vfunc[6]:	codecvt<char, char, __mbstate_t>::do_always_noconv() const
vfunc[7]:	See architecture specific part.
vfunc[8]:	codecvt<char, char, __mbstate_t>::do_max_length() const

The Run Time Type Information for the std::codecvt_byname<char, char, __mbstate_t> class is described by Table 16-306

Table 16-306 typeinfo for codecvt_byname<char, char, __mbstate_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for codecvt_byname<char, char, __mbstate_t>

16.1.106.2 Interfaces for Class codecvt_byname<char, char, __mbstate_t>

An LSB conforming implementation shall provide the generic methods for Class std::codecvt_byname<char, char, __mbstate_t> specified in Table 16-307, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-307 libstdcxx - Class codecvt_byname<char, char, __mbstate_t> Function Interfaces

codecvt_byname<char, char, __mbstate_t>::~codecvt_byname()(GLIBCXX_3.4) [ISOCXX]
codecvt_byname<char, char, __mbstate_t>::~codecvt_byname()(GLIBCXX_3.4) [ISOCXX]
codecvt_byname<char, char, __mbstate_t>::~codecvt_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::codecvt_byname<char, char, __mbstate_t> specified in Table 16-308, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-308 libstdcxx - Class codecvt_byname<char, char, __mbstate_t> Data Interfaces

typeinfo for codecvt_byname<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for codecvt_byname<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for codecvt_byname<char, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.107 Class codecvt_byname<wchar_t, char, __mbstate_t>

16.1.107.1 Class data for codecvt_byname<wchar_t, char, __mbstate_t>

The virtual table for the std::codecvt_byname<wchar_t, char, __mbstate_t> class is described by Table 16-309

Table 16-309 Primary vtable for codecvt_byname<wchar_t, char, __mbstate_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for codecvt_byname<wchar_t, char, __mbstate_t>
vfunc[0]:	codecvt_byname<wchar_t, char, __mbstate_t>::~codecvt_byname()
vfunc[1]:	codecvt_byname<wchar_t, char, __mbstate_t>::~codecvt_byname()
vfunc[2]:	codecvt<wchar_t, char, __mbstate_t>::do_out(__mbstate_t&, wchar_t const*, wchar_t const*, wchar_t const*&, char*, char*, char*&) const
vfunc[3]:	codecvt<wchar_t, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const
vfunc[4]:	codecvt<wchar_t, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, wchar_t*, wchar_t*, wchar_t*&) const
vfunc[5]:	codecvt<wchar_t, char, __mbstate_t>::do_encoding() const
vfunc[6]:	codecvt<wchar_t, char, __mbstate_t>::do_always_noconv() const
vfunc[7]:	See architecture specific part.
vfunc[8]:	codecvt<wchar_t, char, __mbstate_t>::do_max_length() const

The Run Time Type Information for the std::codecvt_byname<wchar_t, char, __mbstate_t> class is described by Table 16-310

Table 16-310 typeinfo for codecvt_byname<wchar_t, char, __mbstate_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for codecvt_byname<wchar_t, char, __mbstate_t>

16.1.107.2 Interfaces for Class codecvt_byname<wchar_t, char, __mbstate_t>

An LSB conforming implementation shall provide the generic methods for Class std::codecvt_byname<wchar_t, char, __mbstate_t> specified in Table 16-311, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-311 libstdcxx - Class codecvt_byname<wchar_t, char, __mbstate_t> Function Interfaces

codecvt_byname<wchar_t, char, __mbstate_t>::~codecvt_byname()(GLIBCXX_3.4) [ISOCXX]
codecvt_byname<wchar_t, char, __mbstate_t>::~codecvt_byname()(GLIBCXX_3.4) [ISOCXX]
codecvt_byname<wchar_t, char, __mbstate_t>::~codecvt_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::codecvt_byname<wchar_t, char, __mbstate_t> specified in Table 16-312, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-312 libstdcxx - Class codecvt_byname<wchar_t, char, __mbstate_t> Data Interfaces

typeinfo for codecvt_byname<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for codecvt_byname<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for codecvt_byname<wchar_t, char, __mbstate_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.108 Class collate<char>

16.1.108.1 Class data for collate<char>

The virtual table for the std::collate<char> class is described by Table 16-313

Table 16-313 Primary vtable for collate<char>

Base Offset	0

Virtual Base Offset	0
RTTI	typeinfo for collate<char>
vfunc[0]:	collate<char>::~collate()
vfunc[1]:	collate<char>::~collate()
vfunc[2]:	collate<char>::do_compare(char const*, char const*, char const*, char const*) const
vfunc[3]:	collate<char>::do_transform(char const*, char const*) const
vfunc[4]:	collate<char>::do_hash(char const*, char const*) const

The Run Time Type Information for the std::collate<char> class is described by Table 16-314

Table 16-314 typeinfo for collate<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for collate<char>

16.1.108.2 Interfaces for Class collate<char>

An LSB conforming implementation shall provide the generic methods for Class std::collate<char> specified in Table 16-315, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-315 libstdcxx - Class collate<char> Function Interfaces

collate<char>::_M_compare(char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::do_compare(char const*, char const*, char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::do_transform(char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::hash(char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::compare(char const*, char const*, char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::do_hash(char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::transform(char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::~collate()(GLIBCXX_3.4) [ISOCXX]

collate<char>::~collate()(GLIBCXX_3.4) [ISOCXX]
collate<char>::~collate()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::collate<char> specified in Table 16-316, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-316 libstdcxx - Class collate<char> Data Interfaces

guard variable for collate<char>::id(GLIBCXX_3.4) [CXXABI-1.86]
collate<char>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for collate<char>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for collate<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for collate<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.109 Class collate<wchar_t>

16.1.109.1 Class data for collate<wchar_t>

The virtual table for the std::collate<wchar_t> class is described by Table 16-317

Table 16-317 Primary vtable for collate<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for collate<wchar_t>
vfunc[0]:	collate<wchar_t>::~collate()
vfunc[1]:	collate<wchar_t>::~collate()
vfunc[2]:	collate<wchar_t>::do_compare(wchar_t const*, wchar_t const*, wchar_t const*, wchar_t const*) const
vfunc[3]:	collate<wchar_t>::do_transform(wchar_t const*, wchar_t const*) const
vfunc[4]:	collate<wchar_t>::do_hash(wchar_t const*, wchar_t const*) const

The Run Time Type Information for the std::collate<wchar_t> class is described by Table 16-318

Table 16-318 typeinfo for collate<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for collate<wchar_t>

16.1.109.2 Interfaces for Class collate<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::collate<wchar_t> specified in Table 16-319, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-319 libstdcxx - Class collate<wchar_t> Function Interfaces

collate<wchar_t>::_M_compare(wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::do_compare(wchar_t const*, wchar_t const*, wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::do_transform(wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::hash(wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::compare(wchar_t const*, wchar_t const*, wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::do_hash(wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::transform(wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::~collate()(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::~collate()(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t>::~collate()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::collate<wchar_t> specified in Table 16-320, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-320 libstdcxx - Class collate<wchar_t> Data Interfaces

guard variable for collate<wchar_t>::id(GLIBCXX_3.4) [CXXABI-1.86]
collate<wchar_t>::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for collate<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for collate<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for collate<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.110 Class collate_byname<char>

16.1.110.1 Class data for collate_byname<char>

The virtual table for the std::collate_byname<char> class is described by Table 16-321

Table 16-321 Primary vtable for collate_byname<char>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for collate_byname<char>
vfunc[0]:	collate_byname<char>::~collate_byname()
vfunc[1]:	collate_byname<char>::~collate_byname()
vfunc[2]:	collate<char>::do_compare(char const*, char const*, char const*, char const*) const
vfunc[3]:	collate<char>::do_transform(char const*, char const*) const
vfunc[4]:	collate<char>::do_hash(char const*, char const*) const

The Run Time Type Information for the std::collate_byname<char> class is described by Table 16-322

Table 16-322 typeinfo for collate_byname<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for collate_byname<char>

16.1.110.2 Interfaces for Class collate_byname<char>

An LSB conforming implementation shall provide the generic methods for Class std::collate_byname<char> specified in Table 16-323, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-323 libstdcxx - Class collate_byname<char> Function Interfaces

collate_byname<char>::~collate_byname()(GLIBCXX_3.4) [ISOCXX]
collate_byname<char>::~collate_byname()(GLIBCXX_3.4) [ISOCXX]
collate_byname<char>::~collate_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::collate_byname<char> specified in Table 16-324, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-324 libstdcxx - Class collate_byname<char> Data Interfaces

typeinfo for collate_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for collate_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for collate_byname<char>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.111 Class collate_byname<wchar_t>

16.1.111.1 Class data for collate_byname<wchar_t>

The virtual table for the std::collate_byname<wchar_t> class is described by Table 16-325

Table 16-325 Primary vtable for collate_byname<wchar_t>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for collate_byname<wchar_t>
vfunc[0]:	collate_byname<wchar_t>::~collate_byname()
vfunc[1]:	collate_byname<wchar_t>::~collate_byname()
vfunc[2]:	collate<wchar_t>::do_compare(wchar_t const*, wchar_t const*, wchar_t const*, wchar_t const*) const
vfunc[3]:	collate<wchar_t>::do_transform(wchar_t const*, wchar_t const*) const
vfunc[4]:	collate<wchar_t>::do_hash(wchar_t const*, wchar_t const*) const

The Run Time Type Information for the std::collate_byname<wchar_t> class is described by Table 16-326

Table 16-326 typeinfo for collate_byname<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for collate_byname<wchar_t>

16.1.111.2 Interfaces for Class collate_byname<wchar_t>

An LSB conforming implementation shall provide the generic methods for Class std::collate_byname<wchar_t> specified in Table 16-327, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-327 libstdcxx - Class collate_byname<wchar_t> Function Interfaces

collate_byname<wchar_t>::~collate_byname()(GLIBCXX_3.4) [ISOCXX]

collate_byname<wchar_t>::~collate_byname()(GLIBCXX_3.4) [ISOCXX]
collate_byname<wchar_t>::~collate_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::collate_byname<wchar_t> specified in Table 16-328, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-328 libstdcxx - Class collate_byname<wchar_t> Data Interfaces

typeinfo for collate_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for collate_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for collate_byname<wchar_t>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.112 Class time_base

16.1.112.1 Class data for time_base

The Run Time Type Information for the std::time_base class is described by Table 16-329

Table 16-329 typeinfo for time_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for time_base

16.1.112.2 Interfaces for Class time_base

No external methods are defined for libstdcxx - Class std::time_base in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_base specified in Table 16-330, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-330 libstdcxx - Class time_base Data Interfaces

typeinfo for time_base(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_base(GLIBCXX_3.4) [CXXABI-1.86]

16.1.113 Class time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >

16.1.113.1 Class data for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-331

Table 16-331 Primary vtable for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >::~time_get_byname()
vfunc[1]:	time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >::~time_get_byname()
vfunc[2]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_date_order() const
vfunc[3]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_time(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[4]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_date(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[5]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_weekday(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const

vfunc[6]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_monthname(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[7]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_year(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const

The Run Time Type Information for the std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-332

Table 16-332 typeinfo for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >

16.1.113.2 Interfaces for Class time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-333, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-333 libstdcxx - Class time_get_byname<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces

time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >::~time_get_byname()(GLIBCXX_3.4) [ISOCXX]
time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >::~time_get_byname()(GLIBCXX_3.4) [ISOCXX]
time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >::~time_get_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-334, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-334 libstdcxx - Class time_get_byname<char, istreambuf_iterator<char, char_traits<char> > > Data Interfaces

typeinfo for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.114 Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.114.1 Class data for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-335

Table 16-335 Primary vtable for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get_byname()
vfunc[1]:	time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get_byname()
vfunc[2]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_date_order() const

vfunc[3]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_time(istreambuf_iterator< wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[4]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_date(istreambuf_iterator<w char_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[5]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_weekday(istreambuf_iterat or<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[6]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_monthname(istreambuf_ite rator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[7]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_year(istreambuf_iterator< wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const

The Run Time Type Information for the std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-336

Table 16-336 typeinfo for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.114.2 Interfaces for Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-337, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-337 libstdcxx - Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get_byname()(GLIBCXX_3.4) [ISOCXX]
time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get_byname()(GLIBCXX_3.4) [ISOCXX]
time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-338, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-338 libstdcxx - Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

typeinfo for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.115 Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >

16.1.115.1 Class data for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-339

Table 16-339 Primary vtable for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put_byname()
vfunc[1]:	time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put_byname()
vfunc[2]:	time_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, tm const*, char, char) const

The Run Time Type Information for the std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-340

Table 16-340 typeinfo for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >

16.1.115.2 Interfaces for Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-341, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-341 libstdcxx - Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces

time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put_byname()(GLIBCXX_3.4) [ISOCXX]
time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put_byname()(GLIBCXX_3.4) [ISOCXX]
time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-342, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-342 libstdcxx - Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > > Data Interfaces

typeinfo for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.116 Class time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.116.1 Class data for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-343

Table 16-343 Primary vtable for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0
Virtual Base Offset	0

RTTI	typeinfo for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put_byname()
vfunc[1]:	time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put_byname()
vfunc[2]:	time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, tm const*, char, char) const

The Run Time Type Information for the std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-344

Table 16-344 typeinfo for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.116.2 Interfaces for Class time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-345, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-345 libstdcxx - Class time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put_byname()(GLIBCXX_3.4) [ISOCXX]
time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put_byname()(GLIBCXX_3.4) [ISOCXX]

time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-346, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-346 libstdcxx - Class time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

typeinfo for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.117 Class time_get<char, istreambuf_iterator<char, char_traits<char> > >

16.1.117.1 Class data for time_get<char, istreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::time_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-347

Table 16-347 Primary vtable for time_get<char, istreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for time_get<char, istreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::~time_get()
vfunc[1]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::~time_get()
vfunc[2]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_date_order() const
vfunc[3]:	time_get<char, istreambuf_iterator<char,

	char_traits<char> > >::do_get_time(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[4]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_date(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[5]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_weekday(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[6]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_monthname(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[7]:	time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_year(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const

16.1.117.2 Interfaces for Class time_get<char, istreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-348, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-348 libstdcxx - Class time_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces

time_get<char, istreambuf_iterator<char, char_traits<char> > >::date_order() const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_date(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_time(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_year(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::get_weekday(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_date_order() const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::get_monthname(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_weekday(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get_monthname(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_via_format(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*, char const*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::get_date(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]

time_get<char, istreambuf_iterator<char, char_traits<char> > >::get_time(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::get_year(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::~time_get()(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::~time_get()(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::~time_get()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-349, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-349 libstdcxx - Class time_get<char, istreambuf_iterator<char, char_traits<char> > > Data Interfaces

guard variable for time_get<char, istreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
time_get<char, istreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for time_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.118 Class time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.118.1 Class data for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::time_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-350

Table 16-350 Primary vtable for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0

Virtual Base Offset	0
RTTI	typeinfo for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get()
vfunc[1]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get()
vfunc[2]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_date_order() const
vfunc[3]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_time(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[4]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_date(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[5]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_weekday(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[6]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_monthname(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t,

	char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const
vfunc[7]:	time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_year(istreambuf_iterator< wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const

16.1.118.2 Interfaces for Class time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-351, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-351 libstdcxx - Class time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::date_order() const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_date(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_time(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_year(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get_weekday(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_date_order() const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get_monthname(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]

time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_weekday(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get_monthname(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_via_format(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get_date(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get_time(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get_year(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, tm*) const(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get()(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get()(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_get()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-352, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-352 libstdcxx - Class time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

guard variable for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.119 Class time_put<char, ostreambuf_iterator<char, char_traits<char> > >

16.1.119.1 Class data for time_put<char, ostreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::time_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-353

Table 16-353 Primary vtable for time_put<char, ostreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for time_put<char, ostreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	time_put<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put()
vfunc[1]:	time_put<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put()
vfunc[2]:	time_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, tm const*, char, char) const

The Run Time Type Information for the std::time_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-354

Table 16-354 typeinfo for time_put<char, ostreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info	
Name	typeinfo name for time_put<char,	

	ostreambuf_iterator<char, char_traits<char> > >	
flags:	8	
basetype:	typeinfo for locale::facet	2
basetype:	typeinfo for time_base	2

16.1.119.2 Interfaces for Class time_put<char, ostreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-355, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-355 libstdcxx - Class time_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces

time_put<char, ostreambuf_iterator<char, char_traits<char> > >::put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, tm const*, char const*, char const*) const(GLIBCXX_3.4) [ISOCXX]
time_put<char, ostreambuf_iterator<char, char_traits<char> > >::put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, tm const*, char, char) const(GLIBCXX_3.4) [ISOCXX]
time_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, tm const*, char, char) const(GLIBCXX_3.4) [ISOCXX]
time_put<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put()(GLIBCXX_3.4) [ISOCXX]
time_put<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put()(GLIBCXX_3.4) [ISOCXX]
time_put<char, ostreambuf_iterator<char, char_traits<char> > >::~time_put()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-356, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-356 libstdcxx - Class time_put<char, ostreambuf_iterator<char, char_traits<char> > > Data Interfaces

guard variable for time_put<char, ostreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
time_put<char, ostreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [ISOCXX]

typeinfo for time_put<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_put<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_put<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.120 Class time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.120.1 Class data for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::time_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-357

Table 16-357 Primary vtable for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put()
vfunc[1]:	time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put()
vfunc[2]:	time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, tm const*, char, char) const

The Run Time Type Information for the std::time_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-358

Table 16-358 typeinfo for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info	

Name	typeinfo name for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >	
flags:	8	
basetype:	typeinfo for locale::facet	2
basetype:	typeinfo for time_base	2

16.1.120.2 Interfaces for Class time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::time_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-359, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-359 libstdcxx - Class time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, tm const*, wchar_t const*, wchar_t const*) const(GLIBCXX_3.4) [ISOCXX]
time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, tm const*, char, char) const(GLIBCXX_3.4) [ISOCXX]
time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, tm const*, char, char) const(GLIBCXX_3.4) [ISOCXX]
time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put()(GLIBCXX_3.4) [ISOCXX]
time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put()(GLIBCXX_3.4) [ISOCXX]
time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~time_put()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::time_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-360, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-360 libstdcxx - Class time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

guard variable for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.121 Class moneypunct<char, false>

16.1.121.1 Class data for moneypunct<char, false>

The virtual table for the std::moneypunct<char, false> class is described by Table 16-361

Table 16-361 Primary vtable for moneypunct<char, false>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct<char, false>
vfunc[0]:	moneypunct<char, false>::~moneypunct()
vfunc[1]:	moneypunct<char, false>::~moneypunct()
vfunc[2]:	moneypunct<char, false>::do_decimal_point() const
vfunc[3]:	moneypunct<char, false>::do_thousands_sep() const
vfunc[4]:	moneypunct<char, false>::do_grouping() const
vfunc[5]:	moneypunct<char, false>::do_curr_symbol() const
vfunc[6]:	moneypunct<char, false>::do_positive_sign() const
vfunc[7]:	moneypunct<char, false>::do_negative_sign() const

vfunc[8]:	moneypunct<char, false>::do_frac_digits() const
vfunc[9]:	moneypunct<char, false>::do_pos_format() const
vfunc[10]:	moneypunct<char, false>::do_neg_format() const

16.1.121.2 Interfaces for Class moneypunct<char, false>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct<char, false> specified in Table 16-362, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-362 libstdcxx - Class moneypunct<char, false> Function Interfaces

moneypunct<char, false>::neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::do_thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::_M_initialize_moneypunct(__locale_struct*, char const*)(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]

moneypunct<char, false>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct<char, false> specified in Table 16-363, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-363 libstdcxx - Class moneypunct<char, false> Data Interfaces

guard variable for moneypunct<char, false>::id(GLIBCXX_3.4) [CXXABI-1.86]
moneypunct<char, false>::id(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct<char, false>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct<char, false>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct<char, false>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.122 Class moneypunct<char, true>

16.1.122.1 Class data for moneypunct<char, true>

The virtual table for the std::moneypunct<char, true> class is described by Table 16-364

Table 16-364 Primary vtable for moneypunct<char, true>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct<char, true>
vfunc[0]:	moneypunct<char, true>::~moneypunct()
vfunc[1]:	moneypunct<char, true>::~moneypunct()
vfunc[2]:	moneypunct<char, true>::do_decimal_point() const
vfunc[3]:	moneypunct<char, true>::do_thousands_sep() const
vfunc[4]:	moneypunct<char, true>::do_grouping() const
vfunc[5]:	moneypunct<char, true>::do_curr_symbol() const
vfunc[6]:	moneypunct<char, true>::do_positive_sign() const
vfunc[7]:	moneypunct<char, true>::do_negative_sign() const

vfunc[8]:	moneypunct<char, true>::do_frac_digits() const
vfunc[9]:	moneypunct<char, true>::do_pos_format() const
vfunc[10]:	moneypunct<char, true>::do_neg_format() const

16.1.122.2 Interfaces for Class moneypunct<char, true>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct<char, true> specified in Table 16-365, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-365 libstdcxx - Class moneypunct<char, true> Function Interfaces

moneypunct<char, true>::neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::do_thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::_M_initialize_moneypunct(__locale_struct*, char const*)(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct<char, true> specified in Table 16-366, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-366 libstdcxx - Class moneypunct<char, true> Data Interfaces

guard variable for moneypunct<char, true>::id(GLIBCXX_3.4) [CXXABI-1.86]
moneypunct<char, true>::id(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct<char, true>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct<char, true>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct<char, true>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.123 Class moneypunct<wchar_t, false>

16.1.123.1 Class data for moneypunct<wchar_t, false>

The virtual table for the std::moneypunct<wchar_t, false> class is described by Table 16-367

Table 16-367 Primary vtable for moneypunct<wchar_t, false>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct<wchar_t, false>
vfunc[0]:	moneypunct<wchar_t, false>::~moneypunct()
vfunc[1]:	moneypunct<wchar_t, false>::~moneypunct()
vfunc[2]:	moneypunct<wchar_t, false>::do_decimal_point() const
vfunc[3]:	moneypunct<wchar_t, false>::do_thousands_sep() const
vfunc[4]:	moneypunct<wchar_t, false>::do_grouping() const
vfunc[5]:	moneypunct<wchar_t, false>::do_curr_symbol() const
vfunc[6]:	moneypunct<wchar_t, false>::do_positive_sign() const
vfunc[7]:	moneypunct<wchar_t, false>::do_negative_sign() const

vfunc[8]:	moneypunct<wchar_t, false>::do_frac_digits() const
vfunc[9]:	moneypunct<wchar_t, false>::do_pos_format() const
vfunc[10]:	moneypunct<wchar_t, false>::do_neg_format() const

16.1.123.2 Interfaces for Class moneypunct<wchar_t, false>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct<wchar_t, false> specified in Table 16-368, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-368 libstdcxx - Class moneypunct<wchar_t, false> Function Interfaces

moneypunct<wchar_t, false>::neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::do_thousands_sep() const(GLIBCXX_3.4) [ISOCXX]

moneypunct<wchar_t, false>::grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::_M_initialize_moneypunct(__locale_struct*, char const*)(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct<wchar_t, false> specified in Table 16-369, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-369 libstdcxx - Class moneypunct<wchar_t, false> Data Interfaces

guard variable for moneypunct<wchar_t, false>::id(GLIBCXX_3.4) [CXXABI-1.86]
moneypunct<wchar_t, false>::id(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct<wchar_t, false>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct<wchar_t, false>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct<wchar_t, false>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.124 Class moneypunct<wchar_t, true>

16.1.124.1 Class data for moneypunct<wchar_t, true>

The virtual table for the std::moneypunct<wchar_t, true> class is described by Table 16-370

Table 16-370 Primary vtable for moneypunct<wchar_t, true>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct<wchar_t, true>
vfunc[0]:	moneypunct<wchar_t, true>::~moneypunct()
vfunc[1]:	moneypunct<wchar_t, true>::~moneypunct()
vfunc[2]:	moneypunct<wchar_t, true>::do_decimal_point() const
vfunc[3]:	moneypunct<wchar_t, true>::do_thousands_sep() const

vfunc[4]:	moneypunct<wchar_t, true>::do_grouping() const
vfunc[5]:	moneypunct<wchar_t, true>::do_curr_symbol() const
vfunc[6]:	moneypunct<wchar_t, true>::do_positive_sign() const
vfunc[7]:	moneypunct<wchar_t, true>::do_negative_sign() const
vfunc[8]:	moneypunct<wchar_t, true>::do_frac_digits() const
vfunc[9]:	moneypunct<wchar_t, true>::do_pos_format() const
vfunc[10]:	moneypunct<wchar_t, true>::do_neg_format() const

16.1.124.2 Interfaces for Class moneypunct<wchar_t, true>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct<wchar_t, true> specified in Table 16-371, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-371 libstdcxx - Class moneypunct<wchar_t, true> Function Interfaces

moneypunct<wchar_t, true>::neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::decimal_point() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_neg_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_pos_format() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_curr_symbol() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_frac_digits() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_decimal_point() const(GLIBCXX_3.4) [ISOCXX]

moneypunct<wchar_t, true>::do_negative_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_positive_sign() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::do_thousands_sep() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::grouping() const(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::_M_initialize_moneypunct(__locale_struct*, char const*)(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::~moneypunct()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct<wchar_t, true> specified in Table 16-372, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-372 libstdcxx - Class moneypunct<wchar_t, true> Data Interfaces

guard variable for moneypunct<wchar_t, true>::id(GLIBCXX_3.4) [CXXABI-1.86]
moneypunct<wchar_t, true>::id(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct<wchar_t, true>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct<wchar_t, true>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct<wchar_t, true>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.125 Class moneypunct_byname<char, false>

16.1.125.1 Class data for moneypunct_byname<char, false>

The virtual table for the std::moneypunct_byname<char, false> class is described by Table 16-373

Table 16-373 Primary vtable for moneypunct_byname<char, false>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct_byname<char, false>
vfunc[0]:	moneypunct_byname<char, false>::~moneypunct_byname()

vfunc[1]:	moneypunct_byname<char, false>::~moneypunct_byname()
vfunc[2]:	moneypunct<char, false>::do_decimal_point() const
vfunc[3]:	moneypunct<char, false>::do_thousands_sep() const
vfunc[4]:	moneypunct<char, false>::do_grouping() const
vfunc[5]:	moneypunct<char, false>::do_curr_symbol() const
vfunc[6]:	moneypunct<char, false>::do_positive_sign() const
vfunc[7]:	moneypunct<char, false>::do_negative_sign() const
vfunc[8]:	moneypunct<char, false>::do_frac_digits() const
vfunc[9]:	moneypunct<char, false>::do_pos_format() const
vfunc[10]:	moneypunct<char, false>::do_neg_format() const

The Run Time Type Information for the std::moneypunct_byname<char, false> class is described by Table 16-374

Table 16-374 typeinfo for moneypunct_byname<char, false>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for moneypunct_byname<char, false>

16.1.125.2 Interfaces for Class moneypunct_byname<char, false>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct_byname<char, false> specified in Table 16-375, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-375 libstdcxx - Class moneypunct_byname<char, false> Function Interfaces

moneypunct_byname<char, false>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<char, false>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]

moneypunct_byname<char, false>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct_byname<char, false> specified in Table 16-376, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-376 libstdcxx - Class moneypunct_byname<char, false> Data Interfaces

moneypunct_byname<char, false>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct_byname<char, false>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct_byname<char, false>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct_byname<char, false>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.126 Class moneypunct_byname<char, true>

16.1.126.1 Class data for moneypunct_byname<char, true>

The virtual table for the std::moneypunct_byname<char, true> class is described by Table 16-377

Table 16-377 Primary vtable for moneypunct_byname<char, true>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct_byname<char, true>
vfunc[0]:	moneypunct_byname<char, true>::~moneypunct_byname()
vfunc[1]:	moneypunct_byname<char, true>::~moneypunct_byname()
vfunc[2]:	moneypunct<char, true>::do_decimal_point() const
vfunc[3]:	moneypunct<char, true>::do_thousands_sep() const
vfunc[4]:	moneypunct<char, true>::do_grouping() const
vfunc[5]:	moneypunct<char, true>::do_curr_symbol() const
vfunc[6]:	moneypunct<char, true>::do_positive_sign() const

vfunc[7]:	moneypunct<char, true>::do_negative_sign() const
vfunc[8]:	moneypunct<char, true>::do_frac_digits() const
vfunc[9]:	moneypunct<char, true>::do_pos_format() const
vfunc[10]:	moneypunct<char, true>::do_neg_format() const

The Run Time Type Information for the std::moneypunct_byname<char, true> class is described by Table 16-378

Table 16-378 typeinfo for moneypunct_byname<char, true>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for moneypunct_byname<char, true>

16.1.126.2 Interfaces for Class moneypunct_byname<char, true>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct_byname<char, true> specified in Table 16-379, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-379 libstdcxx - Class moneypunct_byname<char, true> Function Interfaces

moneypunct_byname<char, true>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<char, true>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<char, true>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct_byname<char, true> specified in Table 16-380, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-380 libstdcxx - Class moneypunct_byname<char, true> Data Interfaces

moneypunct_byname<char, true>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct_byname<char, true>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct_byname<char, true>(GLIBCXX_3.4) [CXXABI-1.86]

vtable for moneypunct_byname<char, true>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.127 Class moneypunct_byname<wchar_t, false>

16.1.127.1 Class data for moneypunct_byname<wchar_t, false>

The virtual table for the std::moneypunct_byname<wchar_t, false> class is described by Table 16-381

Table 16-381 Primary vtable for moneypunct_byname<wchar_t, false>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct_byname<wchar_t, false>
vfunc[0]:	moneypunct_byname<wchar_t, false>::~moneypunct_byname()
vfunc[1]:	moneypunct_byname<wchar_t, false>::~moneypunct_byname()
vfunc[2]:	moneypunct<wchar_t, false>::do_decimal_point() const
vfunc[3]:	moneypunct<wchar_t, false>::do_thousands_sep() const
vfunc[4]:	moneypunct<wchar_t, false>::do_grouping() const
vfunc[5]:	moneypunct<wchar_t, false>::do_curr_symbol() const
vfunc[6]:	moneypunct<wchar_t, false>::do_positive_sign() const
vfunc[7]:	moneypunct<wchar_t, false>::do_negative_sign() const
vfunc[8]:	moneypunct<wchar_t, false>::do_frac_digits() const
vfunc[9]:	moneypunct<wchar_t, false>::do_pos_format() const
vfunc[10]:	moneypunct<wchar_t, false>::do_neg_format() const

The Run Time Type Information for the std::moneypunct_byname<wchar_t, false> class is described by Table 16-382

Table 16-382 typeinfo for moneypunct_byname<wchar_t, false>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for moneypunct_byname<wchar_t, false>

16.1.127.2 Interfaces for Class moneypunct_byname<wchar_t, false>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct_byname<wchar_t, false> specified in Table 16-383, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-383 libstdcxx - Class moneypunct_byname<wchar_t, false> Function Interfaces

moneypunct_byname<wchar_t, false>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<wchar_t, false>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<wchar_t, false>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct_byname<wchar_t, false> specified in Table 16-384, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-384 libstdcxx - Class moneypunct_byname<wchar_t, false> Data Interfaces

moneypunct_byname<wchar_t, false>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct_byname<wchar_t, false>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct_byname<wchar_t, false>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct_byname<wchar_t, false>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.128 Class moneypunct_byname<wchar_t, true>

16.1.128.1 Class data for moneypunct_byname<wchar_t, true>

The virtual table for the std::moneypunct_byname<wchar_t, true> class is described by Table 16-385

Table 16-385 Primary vtable for moneypunct_byname<wchar_t, true>

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for moneypunct_byname<wchar_t, true>
vfunc[0]:	moneypunct_byname<wchar_t, true>::~moneypunct_byname()
vfunc[1]:	moneypunct_byname<wchar_t, true>::~moneypunct_byname()
vfunc[2]:	moneypunct<wchar_t, true>::do_decimal_point() const
vfunc[3]:	moneypunct<wchar_t, true>::do_thousands_sep() const
vfunc[4]:	moneypunct<wchar_t, true>::do_grouping() const
vfunc[5]:	moneypunct<wchar_t, true>::do_curr_symbol() const
vfunc[6]:	moneypunct<wchar_t, true>::do_positive_sign() const
vfunc[7]:	moneypunct<wchar_t, true>::do_negative_sign() const
vfunc[8]:	moneypunct<wchar_t, true>::do_frac_digits() const
vfunc[9]:	moneypunct<wchar_t, true>::do_pos_format() const
vfunc[10]:	moneypunct<wchar_t, true>::do_neg_format() const

The Run Time Type Information for the std::moneypunct_byname<wchar_t, true> class is described by Table 16-386

Table 16-386 typeinfo for moneypunct_byname<wchar_t, true>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for moneypunct_byname<wchar_t, true>

16.1.128.2 Interfaces for Class moneypunct_byname<wchar_t, true>

An LSB conforming implementation shall provide the generic methods for Class std::moneypunct_byname<wchar_t, true> specified in Table 16-387, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-387 libstdcxx - Class moneypunct_byname<wchar_t, true> Function Interfaces

moneypunct_byname<wchar_t, true>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<wchar_t, true>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]
moneypunct_byname<wchar_t, true>::~moneypunct_byname()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::moneypunct_byname<wchar_t, true> specified in Table 16-388, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-388 libstdcxx - Class moneypunct_byname<wchar_t, true> Data Interfaces

moneypunct_byname<wchar_t, true>::intl(GLIBCXX_3.4) [ISOCXX]
typeinfo for moneypunct_byname<wchar_t, true>(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for moneypunct_byname<wchar_t, true>(GLIBCXX_3.4) [CXXABI-1.86]
vtable for moneypunct_byname<wchar_t, true>(GLIBCXX_3.4) [CXXABI-1.86]

16.1.129 Class money_base

16.1.129.1 Class data for money_base

The Run Time Type Information for the std::money_base class is described by Table 16-389

Table 16-389 typeinfo for money_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for money_base

16.1.129.2 Interfaces for Class money_base

An LSB conforming implementation shall provide the generic methods for Class std::money_base specified in Table 16-390, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-390 libstdcxx - Class money_base Function Interfaces

money_base::_S_construct_pattern(char, char, char)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::money_base specified in Table 16-391, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-391 libstdcxx - Class money_base Data Interfaces

money_base::_S_default_pattern(GLIBCXX_3.4) [ISOCXX]
money_base::_S_atoms(GLIBCXX_3.4) [ISOCXX]
typeinfo for money_base(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for money_base(GLIBCXX_3.4) [CXXABI-1.86]

16.1.130 Class money_get<char, istreambuf_iterator<char, char_traits<char> > >

16.1.130.1 Class data for money_get<char, istreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-392

Table 16-392 Primary vtable for money_get<char, istreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for money_get<char, istreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	money_get<char, istreambuf_iterator<char, char_traits<char> > >::~money_get()
vfunc[1]:	money_get<char, istreambuf_iterator<char, char_traits<char> > >::~money_get()
vfunc[2]:	money_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char,

	char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, bool, ios_base&, _Ios_Iostate&, long double&) const
vfunc[3]:	money_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, bool, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const

The Run Time Type Information for the std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-393

Table 16-393 typeinfo for money_get<char, istreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for money_get<char, istreambuf_iterator<char, char_traits<char> > >

16.1.130.2 Interfaces for Class money_get<char, istreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-394, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-394 libstdcxx - Class money_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces

istreambuf_iterator<char, char_traits<char> > money_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract<false>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<char, char_traits<char> > money_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract<true>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&,

basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, bool, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, bool, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, bool, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, bool, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::~money_get()(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::~money_get()(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::~money_get()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::money_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-395, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-395 libstdcxx - Class money_get<char, istreambuf_iterator<char, char_traits<char> > > Data Interfaces

guard variable for money_get<char, istreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
money_get<char, istreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for money_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for money_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for money_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.131 Class money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.131.1 Class data for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-396

Table 16-396 Primary vtable for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_get()
vfunc[1]:	money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_get()
vfunc[2]:	money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, _Ios_Iostate&, long double&) const
vfunc[3]:	money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, _Ios_Iostate&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&) const

The Run Time Type Information for the std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-397

Table 16-397 typeinfo for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.131.2 Interfaces for Class money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-398, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-398 libstdcxx - Class money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

istreambuf_iterator<wchar_t, char_traits<wchar_t> > money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract<false>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<wchar_t, char_traits<wchar_t> > money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract<true>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, _Ios_Iostate&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&) const(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&,

_Ios_Iostate&, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >&) const(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_get()(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_get()(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_get()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::money_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-399, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-399 libstdcxx - Class money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

guard variable for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.132 Class money_put<char, ostreambuf_iterator<char, char_traits<char> > >

16.1.132.1 Class data for money_put<char, ostreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::money_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-400

Table 16-400 Primary vtable for money_put<char, ostreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0

RTTI	typeinfo for money_put<char, ostreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	money_put<char, ostreambuf_iterator<char, char_traits<char> > >::~money_put()
vfunc[1]:	money_put<char, ostreambuf_iterator<char, char_traits<char> > >::~money_put()
vfunc[2]:	money_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, bool, ios_base&, char, long double) const
vfunc[3]:	money_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, bool, ios_base&, char, basic_string<char, char_traits<char>, allocator<char> > const&) const

The Run Time Type Information for the std::money_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-401

Table 16-401 typeinfo for money_put<char, ostreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for money_put<char, ostreambuf_iterator<char, char_traits<char> > >

16.1.132.2 Interfaces for Class money_put<char, ostreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::money_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-402, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-402 libstdcxx - Class money_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces

money_put<char, ostreambuf_iterator<char, char_traits<char> > >::put(ostreambuf_iterator<char, char_traits<char> >, bool, ios_base&, char, basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::put(ostreambuf_iterator<char, char_traits<char> >, bool, ios_base&, char, long double) const(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, bool, ios_base&, char, basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, bool, ios_base&, char, long double) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<char, char_traits<char> > money_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert<false>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<char, char_traits<char> > money_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert<true>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, basic_string<char, char_traits<char>, allocator<char> > const&) const(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::~money_put()(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::~money_put()(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::~money_put()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::money_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-403, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-403 libstdcxx - Class money_put<char, ostreambuf_iterator<char, char_traits<char> > > Data Interfaces

guard variable for money_put<char, ostreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
money_put<char, ostreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [ISOCXX]

typeinfo for money_put<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for money_put<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for money_put<char, ostreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.133 Class money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.133.1 Class data for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::money_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-404

Table 16-404 Primary vtable for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_put()
vfunc[1]:	money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_put()
vfunc[2]:	money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, wchar_t, long double) const
vfunc[3]:	money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, wchar_t,

	basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const

The Run Time Type Information for the std::money_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-405

Table 16-405 typeinfo for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.133.2 Interfaces for Class money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::money_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-406, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-406 libstdcxx - Class money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, wchar_t, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, wchar_t, long double) const(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, wchar_t, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_put(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, bool, ios_base&, wchar_t, long double) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<wchar_t, char_traits<wchar_t> > money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_insert<false>(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]

ostreambuf_iterator<wchar_t, char_traits<wchar_t> > money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_insert<true>(ostreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, wchar_t, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&) const(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_put()(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_put()(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~money_put()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::money_put<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-407, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-407 libstdcxx - Class money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

guard variable for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.134 Class locale

16.1.134.1 Interfaces for Class locale

An LSB conforming implementation shall provide the generic methods for Class std::locale specified in Table 16-408, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-408 libstdcxx - Class locale Function Interfaces

locale::id::_M_id() const(GLIBCXX_3.4) [ISOCXX]
locale::name() const(GLIBCXX_3.4) [ISOCXX]
locale::operator==(locale const&) const(GLIBCXX_3.4) [ISOCXX]
locale::_M_coalesce(locale const&, locale const&, int)(GLIBCXX_3.4) [ISOCXX]

locale::_S_normalize_category(int)(GLIBCXX_3.4) [ISOCXX]
locale::_Impl::_M_install_facet(locale::id const*, locale::facet const*)(GLIBCXX_3.4) [LSB]
locale::_Impl::_M_replace_facet(locale::_Impl const*, locale::id const*)(GLIBCXX_3.4) [LSB]
locale::_Impl::~_Impl()(GLIBCXX_3.4) [LSB]
locale::_Impl::~_Impl()(GLIBCXX_3.4) [LSB]
locale::global(locale const&)(GLIBCXX_3.4) [ISOCXX]
locale::classic()(GLIBCXX_3.4) [ISOCXX]
locale::locale(char const*)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale::_Impl*)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale const&)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale const&, locale const&, int)(GLIBCXX_3.4) [ISOCXX]
locale::locale()(GLIBCXX_3.4) [ISOCXX]
locale::locale(char const*)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale::_Impl*)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale const&)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale const&, char const*, int)(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale const&, locale const&, int)(GLIBCXX_3.4) [ISOCXX]
locale::locale()(GLIBCXX_3.4) [ISOCXX]
locale::~locale()(GLIBCXX_3.4) [ISOCXX]
locale::~locale()(GLIBCXX_3.4) [ISOCXX]
locale::operator=(locale const&)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::locale specified in Table 16-409, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-409 libstdcxx - Class locale Data Interfaces

locale::all(GLIBCXX_3.4) [ISOCXX]
locale::none(GLIBCXX_3.4) [ISOCXX]
locale::time(GLIBCXX_3.4) [ISOCXX]
locale::ctype(GLIBCXX_3.4) [ISOCXX]
locale::collate(GLIBCXX_3.4) [ISOCXX]
locale::numeric(GLIBCXX_3.4) [ISOCXX]

locale::messages(GLIBCXX_3.4) [ISOCXX]
locale::monetary(GLIBCXX_3.4) [ISOCXX]

16.1.135 Class locale::facet

16.1.135.1 Class data for locale::facet

The virtual table for the std::locale::facet class is described by Table 16-410

Table 16-410 Primary vtable for locale::facet

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for locale::facet
vfunc[0]:	locale::facet::~facet()
vfunc[1]:	locale::facet::~facet()

The Run Time Type Information for the std::locale::facet class is described by Table 16-411

Table 16-411 typeinfo for locale::facet

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for locale::facet

16.1.135.2 Interfaces for Class locale::facet

An LSB conforming implementation shall provide the generic methods for Class std::locale::facet specified in Table 16-412, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-412 libstdcxx - Class locale::facet Function Interfaces

locale::facet::_S_get_c_name()(GLIBCXX_3.4.6) [ISOCXX]
locale::facet::_S_get_c_locale()(GLIBCXX_3.4) [ISOCXX]
locale::facet::_S_clone_c_locale(__locale_struct*&)(GLIBCXX_3.4) [ISOCXX]
locale::facet::_S_create_c_locale(__locale_struct*&, char const*, __locale_struct*)(GLIBCXX_3.4) [ISOCXX]
locale::facet::_S_destroy_c_locale(__locale_struct*&)(GLIBCXX_3.4) [ISOCXX]
locale::facet::~facet()(GLIBCXX_3.4) [ISOCXX]
locale::facet::~facet()(GLIBCXX_3.4) [ISOCXX]
locale::facet::~facet()(GLIBCXX_3.4) [ISOCXX]
locale::locale(locale const&, char const*, int)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::locale::facet specified in Table 16-413, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-413 libstdcxx - Class locale::facet Data Interfaces

__timepunct_cache<char>::_S_timezones(GLIBCXX_3.4) [ISOCXX]
__timepunct_cache<wchar_t>::_S_timezones(GLIBCXX_3.4) [ISOCXX]
typeinfo for locale::facet(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for locale::facet(GLIBCXX_3.4) [CXXABI-1.86]
vtable for locale::facet(GLIBCXX_3.4) [CXXABI-1.86]

16.1.136 facet functions

16.1.136.1 Interfaces for facet functions

An LSB conforming implementation shall provide the generic methods for facet functions specified in Table 16-414, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-414 libstdcxx - facet functions Function Interfaces

void __convert_to_v<double>(char const*, double&, _Ios_Iostate&, __locale_struct* const&)(GLIBCXX_3.4) [ISOCXX]
void __convert_to_v<long double>(char const*, long double&, _Ios_Iostate&, __locale_struct* const&)(GLIBCXX_3.4) [ISOCXX]
void __convert_to_v<float>(char const*, float&, _Ios_Iostate&, __locale_struct* const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<moneypunct<char, false> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<moneypunct<wchar_t, false> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<ctype<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<codecvt<char, char, __mbstate_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<codecvt<wchar_t, char, __mbstate_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<collate<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<collate<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<num_get<char, istreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]

bool has_facet<num_put<char, ostreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<messages<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<messages<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<numpunct<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<numpunct<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<time_get<char, istreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<time_put<char, ostreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<money_get<char, istreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<money_put<char, ostreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
bool has_facet<money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, false> const& use_facet<moneypunct<char, false> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
moneypunct<char, true> const& use_facet<moneypunct<char, true> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, false> const& use_facet<moneypunct<wchar_t, false> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
moneypunct<wchar_t, true> const& use_facet<moneypunct<wchar_t, true> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
__timepunct<char> const& use_facet<__timepunct<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t> const& use_facet<__timepunct<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
ctype<char> const& use_facet<ctype<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]

ctype<wchar_t> const& use_facet<ctype<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
codecvt<char, char, __mbstate_t> const& use_facet<codecvt<char, char, __mbstate_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
codecvt<wchar_t, char, __mbstate_t> const& use_facet<codecvt<wchar_t, char, __mbstate_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
collate<char> const& use_facet<collate<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
collate<wchar_t> const& use_facet<collate<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > > const& use_facet<num_get<char, istreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > const& use_facet<num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
num_put<char, ostreambuf_iterator<char, char_traits<char> > > const& use_facet<num_put<char, ostreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > const& use_facet<num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
messages<char> const& use_facet<messages<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t> const& use_facet<messages<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
numpunct<char> const& use_facet<numpunct<char> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
numpunct<wchar_t> const& use_facet<numpunct<wchar_t> >(locale const&)(GLIBCXX_3.4) [ISOCXX]
time_get<char, istreambuf_iterator<char, char_traits<char> > > const& use_facet<time_get<char, istreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > const& use_facet<time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
time_put<char, ostreambuf_iterator<char, char_traits<char> > > const& use_facet<time_put<char, ostreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]

time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > const& use_facet<time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
money_get<char, istreambuf_iterator<char, char_traits<char> > > const& use_facet<money_get<char, istreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > const& use_facet<money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
money_put<char, ostreambuf_iterator<char, char_traits<char> > > const& use_facet<money_put<char, ostreambuf_iterator<char, char_traits<char> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]
money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > const& use_facet<money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > >(locale const&)(GLIBCXX_3.4) [ISOCXX]

16.1.137 Class __num_base

16.1.137.1 Class data for __num_base

16.1.137.2 Interfaces for Class __num_base

An LSB conforming implementation shall provide the generic methods for Class std::__num_base specified in Table 16-415, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-415 libstdcxx - Class __num_base Function Interfaces

__num_base::_S_format_float(ios_base const&, char*, char)(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::__num_base specified in Table 16-416, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-416 libstdcxx - Class __num_base Data Interfaces

__num_base::_S_atoms_in(GLIBCXX_3.4) [ISOCXX]
__num_base::_S_atoms_out(GLIBCXX_3.4) [ISOCXX]

16.1.138 Class num_get<char, istreambuf_iterator<char, char_traits<char> > >

16.1.138.1 Class data for num_get<char, istreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::num_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-417

Table 16-417 Primary vtable for num_get<char, istreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for num_get<char, istreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::~num_get()
vfunc[1]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::~num_get()
vfunc[2]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, bool&) const
vfunc[3]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long&) const
vfunc[4]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned short&) const
vfunc[5]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char,

	char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned int&) const
vfunc[6]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long&) const
vfunc[7]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long long&) const
vfunc[8]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long long&) const
vfunc[9]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, float&) const
vfunc[10]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, double&) const
vfunc[11]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char,

	char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long double&) const
vfunc[12]:	num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, void*&) const

The Run Time Type Information for the std::num_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-418

Table 16-418 typeinfo for num_get<char, istreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for num_get<char, istreambuf_iterator<char, char_traits<char> > >
basetype:	typeinfo for locale::facet

16.1.138.2 Interfaces for Class num_get<char, istreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::num_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-419, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-419 libstdcxx - Class num_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces

istreambuf_iterator<char, char_traits<char> > num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_int<unsigned int>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned int&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<char, char_traits<char> > num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_int<long>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long&) const(GLIBCXX_3.4) [ISOCXX]

istreambuf_iterator<char, char_traits<char> > num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_int<unsigned long>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<char, char_traits<char> > num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_int<unsigned short>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned short&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<char, char_traits<char> > num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_int<long long>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long long&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<char, char_traits<char> > num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_int<unsigned long long>(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::_M_extract_float(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, void*&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, bool&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, float&) const(GLIBCXX_3.4) [ISOCXX]

num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned int&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned short&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, void*&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, bool&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, float&) const(GLIBCXX_3.4) [ISOCXX]

num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned int&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned short&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::do_get(istreambuf_iterator<char, char_traits<char> >, istreambuf_iterator<char, char_traits<char> >, ios_base&, _Ios_Iostate&, unsigned long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::~num_get()(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::~num_get()(GLIBCXX_3.4) [ISOCXX]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::~num_get()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::num_get<char, std::istreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-420, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-420 libstdcxx - Class num_get<char, istreambuf_iterator<char, char_traits<char> > > Data Interfaces

guard variable for num_get<char, istreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
num_get<char, istreambuf_iterator<char, char_traits<char> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for num_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

typeinfo name for num_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for num_get<char, istreambuf_iterator<char, char_traits<char> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.139 Class num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

16.1.139.1 Class data for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

The virtual table for the std::num_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-421

Table 16-421 Primary vtable for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >
vfunc[0]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~num_get()
vfunc[1]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~num_get()
vfunc[2]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, bool&) const
vfunc[3]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long&) const
vfunc[4]:	num_get<wchar_t, istreambuf_iterator<wchar_t,

	char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_ t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned short&) const
vfunc[5]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_ t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned int&) const
vfunc[6]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_ t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long&) const
vfunc[7]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_ t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long long&) const
vfunc[8]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_ t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long long&) const
vfunc[9]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_ t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t,

	char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, float&) const
vfunc[10]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, double&) const
vfunc[11]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long double&) const
vfunc[12]:	num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, void*&) const

The Run Time Type Information for the std::num_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > class is described by Table 16-422

Table 16-422 typeinfo for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >
basetype:	typeinfo for locale::facet

16.1.139.2 Interfaces for Class num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >

An LSB conforming implementation shall provide the generic methods for Class std::num_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-423, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-423 libstdcxx - Class num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces

istreambuf_iterator<wchar_t, char_traits<wchar_t> > num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_int<unsigned int>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned int&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<wchar_t, char_traits<wchar_t> > num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_int<long>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<wchar_t, char_traits<wchar_t> > num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_int<unsigned long>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<wchar_t, char_traits<wchar_t> > num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_int<unsigned short>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned short&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<wchar_t, char_traits<wchar_t> > num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_int<long long>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long long&) const(GLIBCXX_3.4) [ISOCXX]
istreambuf_iterator<wchar_t, char_traits<wchar_t> > num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_int<unsigned long long>(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::_M_extract_float(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, basic_string<char, char_traits<char>, allocator<char> >&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, void*&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >,

istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, bool&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, float&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned int&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned short&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, void*&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >,

istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, bool&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long double&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, float&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned int&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned short&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::do_get(istreambuf_iterator<wchar_t, char_traits<wchar_t> >, istreambuf_iterator<wchar_t, char_traits<wchar_t> >, ios_base&, _Ios_Iostate&, unsigned long long&) const(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~num_get()(GLIBCXX_3.4) [ISOCXX]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~num_get()(GLIBCXX_3.4) [ISOCXX]

num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::~num_get()(GLIBCXX_3.4) [ISOCXX]

An LSB conforming implementation shall provide the generic data interfaces for Class std::num_get<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > > specified in Table 16-424, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-424 libstdcxx - Class num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Data Interfaces

guard variable for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [CXXABI-1.86]
num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >::id(GLIBCXX_3.4) [ISOCXX]
typeinfo for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
typeinfo name for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]
vtable for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >(GLIBCXX_3.4) [CXXABI-1.86]

16.1.140 Class num_put<char, ostreambuf_iterator<char, char_traits<char> > >

16.1.140.1 Class data for num_put<char, ostreambuf_iterator<char, char_traits<char> > >

The virtual table for the std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-425

Table 16-425 Primary vtable for num_put<char, ostreambuf_iterator<char, char_traits<char> > >

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for num_put<char, ostreambuf_iterator<char, char_traits<char> > >
vfunc[0]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::~num_put()
vfunc[1]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::~num_put()

vfunc[2]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, bool) const
vfunc[3]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, long) const
vfunc[4]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, unsigned long) const
vfunc[5]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, long long) const
vfunc[6]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, unsigned long long) const
vfunc[7]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, double) const
vfunc[8]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, long double) const
vfunc[9]:	num_put<char, ostreambuf_iterator<char, char_traits<char> > >::do_put(ostreambuf_iterator<char,

	char_traits<char> >, ios_base&, char, void const*) const

The Run Time Type Information for the std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > class is described by Table 16-426

Table 16-426 typeinfo for num_put<char, ostreambuf_iterator<char, char_traits<char> > >

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for num_put<char, ostreambuf_iterator<char, char_traits<char> > >
basetype:	typeinfo for locale::facet

16.1.140.2 Interfaces for Class num_put<char, ostreambuf_iterator<char, char_traits<char> > >

An LSB conforming implementation shall provide the generic methods for Class std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > > specified in Table 16-427, with the full mandatory functionality as described in the referenced underlying specification.

Table 16-427 libstdcxx - Class num_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces

ostreambuf_iterator<char, char_traits<char> > num_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert_int<long>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, long) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<char, char_traits<char> > num_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert_int<unsigned long>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<char, char_traits<char> > num_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert_int<long long>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, long long) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<char, char_traits<char> > num_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert_int<unsigned long long>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, unsigned long long) const(GLIBCXX_3.4) [ISOCXX]
ostreambuf_iterator<char, char_traits<char> > num_put<char, ostreambuf_iterator<char, char_traits<char> > >::_M_insert_float<double>(ostreambuf_iterator<char, char_traits<char> >, ios_base&, char, char, double) const(GLIBCXX_3.4) [ISOCXX]