
Linux Standard Base (LSB) —

Part 3-2:

**Core specification for IA64 (Itanium™)
architecture**

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021



IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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 for IA64 (Itanium™) architecture 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 23360-3-2 cancels and replaces ISO/IEC 23360-3:2006, 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

Foreword	iii
Introduction	vi
I Introductory Elements	1
1 Scope	2
2 References	3
2.1 Normative References	3
2.2 Informative References/Bibliography	5
3 Requirements	8
3.1 Relevant Libraries	8
3.2 LSB Implementation Conformance	8
3.3 LSB Application Conformance	9
4 Terms and Definitions	11
5 Documentation Conventions	13
II Executable and Linking Format (ELF)	14
6 Introduction	15
7 Low Level System Information	16
7.1 Machine Interface	16
7.2 Function Calling Sequence	20
7.3 Operating System Interface	21
7.4 Process Initialization	22
7.5 Coding Examples	24
7.6 C Stack Frame	25
7.7 Debug Information	26
8 Object Format	27
8.1 Introduction	27
8.2 ELF Header	27
8.3 Sections	28
8.4 Symbol Table	30
8.5 Relocation	30
9 Program Loading and Dynamic Linking	31
9.1 Introduction	31
9.2 Program Header	31
9.3 Program Loading	31
9.4 Dynamic Linking	31
III Base Libraries	33
10 Libraries	34
10.1 Program Interpreter/Dynamic Linker	34
10.2 Interfaces for libc	34
10.3 Data Definitions for libc	54
10.4 Interface Definitions for libc	72
10.5 Interfaces for libm	74
10.6 Data Definitions for libm	79
10.7 Interface Definitions for libm	80
10.8 Interfaces for libpthread	81
10.9 Data Definitions for libpthread	86
10.10 Interfaces for libgcc_s	88
10.11 Data Definitions for libgcc_s	88
10.12 Interface Definitions for libgcc_s	89
10.13 Interfaces for libdl	89
10.14 Data Definitions for libdl	90
10.15 Interfaces for libcrypt	90
10.16 Data Definitions for libcrypt	91
IV Utility Libraries	92
11 Libraries	93

11.1 Interfaces for libz	93
11.2 Data Definitions for libz	93
11.3 Interfaces for libncurses	94
11.4 Data Definitions for libncurses	94
11.5 Interfaces for libncursesw	94
11.6 Data Definitions for libncursesw	95
11.7 Interfaces for libutil	95
V Base Libraries.....	97
12 Libraries	98
12.1 Interfaces for libstdcxx.....	98
12.2 Interface Definitions for libstdcxx.....	209
VI Package Format and Installation	210
13 Software Installation.....	211
13.1 Package Dependencies	211
13.2 Package Architecture Considerations.....	211
Annex A Alphabetical Listing of Interfaces by Library.....	212
A.1 libc	212
A.2 libcrypt	227
A.3 libdl	227
A.4 libgcc_s.....	228
A.5 libm	228
A.6 libpthread.....	233
A.7 librt.....	236
A.8 libutil	237

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

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

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

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 Itanium™ architecture specific part of the Core module of the Linux Standard Base (LSB). This part supplements the common part of the LSB Core module with those interfaces that differ between architectures.

This part should be used in conjunction with LSB Core - Generic, the common 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. This part 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
LSB Core - Generic	Linux Standard Base - Core Specification - Generic	http://www.linuxbase.org/spec/
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 3.0	http://refspecs.linuxbase.org/fhs
Intel® Itanium™ Processor-specific Application Binary Interface	Intel® Itanium™ Processor-specific Application Binary Interface	http://refspecs.linuxfoundation.org/elf/IA64-SysV-psABI.pdf
ISO C (1999)	ISO/IEC 9899:1999 - Programming Languages -- C	
ISO/IEC 14882: 2003 C++ Language	ISO/IEC 14882: 2003 Programming languages --C++	
Itanium™ Architecture Software Developer's Manual Volume 1	Itanium™ Architecture Software Developer's Manual Volume 1: Application Architecture	http://refspecs.linuxfoundation.org/IA64-softdevman-vol1.pdf
Itanium™ Architecture Software Developer's Manual Volume 2	Itanium™ Architecture Software Developer's Manual Volume 2: System Architecture	http://refspecs.linuxfoundation.org/IA64-softdevman-vol2.pdf
Itanium™ Architecture Software Developer's Manual Volume 3	Itanium™ Architecture Software Developer's Manual Volume 3: Instruction Set Reference	http://refspecs.linuxfoundation.org/IA64-softdevman-vol3.pdf
Itanium™ Architecture Software Developer's Manual Volume 4	IA-64 Processor Reference: Intel® Itanium™ Processor	http://refspecs.linuxfoundation.org/IA64-softdevman-vol4.pdf

Name	Title	URL
	Reference Manual for Software Development	
Itanium™ C++ ABI	Itanium™ C++ ABI (Revision 1.86)	http://refspecs.linuxfoundation.org/cxxabi-1.86.html
Itanium™ Software Conventions and Runtime Guide	Itanium™ Software Conventions & Runtime Architecture Guide, September 2000	http://refspecs.linuxfoundation.org/IA64conventions.pdf
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.linuxfoundation.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 (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	http://www.unix.org/version3/

Name	Title	URL
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 - DRAFT - 17 December 2003	http://www.sco.com/developers/gabi/2003-12-17/contents.html
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

Name	Title	URL
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 conversion ITUV	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Li18nux Globalization Specification	LI18N 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.openi18n.org/docs/html/LI18N-UX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices-2.6+.txt
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

Name	Title	URL
RFC 1950: ZLIB Compressed Data Format Specification	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/
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/

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on IA64 Linux Standard Base systems, with the specified runtime names. These names override or supplement the names specified in the generic LSB (LSB Core - Generic) specification. The specified program interpreter, referred to as `proginterp` in this table, shall be used to load the shared libraries specified by `DT_NEEDED` entries at run time.

Table 3-1 Standard Library Names

Library	Runtime Name
libc	libc.so.6.1
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libgcc_s	libgcc_s.so.1
libm	libm.so.6.1
libncurses	libncurses.so.5
libncursesw	libncursesw.so.5
libpthread	libpthread.so.0
libstdcxx	libstdc++.so.6
libutil	libutil.so.1
libz	libz.so.1
proginterp	/lib/ld-lsb-ia64.so.3

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.

II Executable and Linking Format (ELF)

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

6 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 Intel® Itanium™ Processor-specific Application Binary Interface, and is intended to document additions made since the publication of that document.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

7 Low Level System Information

7.1 Machine Interface

7.1.1 Processor Architecture

The Itanium™ Architecture is specified by the following documents

- Itanium™ Architecture Software Developer's Manual Volume 1
- Itanium™ Architecture Software Developer's Manual Volume 2
- Itanium™ Architecture Software Developer's Manual Volume 3
- Itanium™ Architecture Software Developer's Manual Volume 4
- Itanium™ Software Conventions and Runtime Guide
- Intel® Itanium™ Processor-specific Application Binary Interface

Only the features of the Itanium™ processor instruction set may be assumed to be present. An application should determine if any additional instruction set features are available before using those additional features. If a feature is not present, then the application may not use it.

Conforming applications may use only instructions which do not require elevated privileges.

Conforming applications shall not invoke the implementations underlying system call interface directly. The interfaces in the implementation base libraries shall be used instead.

Rationale: Implementation-supplied base libraries may use the system call interface but applications must not assume any particular operating system or kernel version is present.

There are some features of the Itanium™ processor architecture that need not be supported by a conforming implementation. These are described in this chapter. A conforming application shall not rely on these features.

Applications conforming to this specification must provide feedback to the user if a feature that is required for correct execution of the application is not present. Applications conforming to this specification should attempt to execute in a diminished capacity if a required feature is not present.

This specification does not provide any performance guarantees of a conforming system. A system conforming to this specification may be implemented in either hardware or software.

This specification describes only LP64 (i.e. 32-bit integers, 64-bit longs and pointers) based implementations. Implementations may also provide ILP32 (32-bit integers, longs, and pointers), but conforming applications shall not rely on support for ILP32. See section 1.2 of the Intel® Itanium™ Processor-specific Application Binary Interface for further information.

7.1.2 Data Representation

The following sections, in conjunction with section 4 of Itanium™ Software Conventions and Runtime Guide, define the size, alignment requirements, and hardware representation of the standard C data types.

Within this specification, the term `byte` refers to an 8-bit object, the term `halfword` refers to a 16-bit object, the term `word` refers to a 32-bit object, the term

doubleword refers to a 64-bit object, and the term quadword refers to a 128-bit object.

7.1.2.1 Byte Ordering

LSB-conforming applications shall use little-endian byte ordering. LSB-conforming implementations may support big-endian applications.

7.1.2.2 Fundamental Types

Table 7-1 describes how fundamental C language data types shall be represented:

Table 7-1 Scalar Types

Type	C	sizeof	Alignment (bytes)	Hardware Representation
Integral	_Bool	1	1	byte (sign unspecified)
	char	1	1	signed byte
	signed char			
	unsigned char			signed byte
	short	2	2	signed half-word
	signed short			
	unsigned short			unsigned halfword
	int	4	4	signed word
	signed int			
	unsigned int			unsigned word
	long	8	8	signed doubleword
	signed long			
	unsigned long			unsigned doubleword
	long long	8	8	signed doubleword
	signed long long			
	unsigned long long			unsigned doubleword
Pointer	<i>any-type</i> *	8	8	unsigned doubleword

Type	C	sizeof	Alignment (bytes)	Hardware Representation
	<i>any-type</i> (*) ()			
Floating-Point	float	4	4	IEEE Single-precision
	double	8	8	IEEE Double-precision
	long double	16	16	IEEE Double-extended

A null pointer (for all types) shall have the value zero.

7.1.2.3 Aggregates and Unions

Aggregates (structures and arrays) and unions assume the alignment of their most strictly aligned component. The size of any object, including aggregates and unions, shall always be a multiple of the object's alignment. An array uses the same alignment as its elements. Structure and union objects may require padding to meet size and element constraints. The contents of such padding is undefined.

- An entire structure or union object shall be aligned on the same boundary as its most strictly aligned member.
- Each member shall be assigned to the lowest available offset with the appropriate alignment. This may require *internal padding*, depending on the previous member.
- A structure's size shall be increased, if necessary, to make it a multiple of the alignment. This may require *tail padding*, depending on the last member.

A conforming application shall not read padding.

<pre>struct { char c; }</pre>	
Byte aligned, sizeof is 1	
Offset	Byte 0
0	c ⁰

Figure 7-1 Structure Smaller Than A Word

<pre>struct { char c; char d; short s; int i; long l; }</pre>
Doubleword Aligned, sizeof is 16

Offset	Byte 3	Byte 2	Byte 1	Byte 0
0	s^2		d^1	c^0
4	i^0			
8	l^0			
12				

Figure 7-2 No Padding

<pre> struct { char c; long l; int i; short s; } </pre>				
Doubleword Aligned, sizeof is 24				
Offset	Byte 3	Byte 2	Byte 1	Byte 0
0	pad^1			c^0
4	pad^1			
8	l^0			
12				
16	i^0			
20	pad^2		s^0	

Figure 7-3 Internal and Tail Padding

7.1.2.4 Bit Fields

C struct and union definitions may have *bit-fields*, which define integral objects with a specified number of bits.

Bit fields that are declared with neither `signed` nor `unsigned` specifier shall always be treated as `unsigned`. Bit fields obey the same size and alignment rules as other structure and union members, with the following additional properties:

- Bit-fields are allocated from right to left (least to most significant).
- A bit-field must entirely reside in a storage unit for its appropriate type. A bit field shall never cross its unit boundary.
- Bit-fields may share a storage unit with other struct/union members, including members that are not bit fields. Such other struct/union members shall occupy different parts of the storage unit.
- The type of unnamed bit-fields shall not affect the alignment of a structure or union, although individual bit-field member offsets shall obey the alignment constraints.

Bit-field Type	Width w	Range
signed char	1 to 8	-2^{w-1} to $2^{w-1}-1$

Bit-field Type	Width w	Range
char unsigned char		0 to 2^w-1 0 to 2^w-1
signed short short unsigned short	1 to 16	-2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1
signed int int unsigned int	1 to 32	-2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1
signed long long unsigned long	1 to 64	-2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1

Figure 7-4 Bit-Field Ranges

7.2 Function Calling Sequence

LSB-conforming applications shall use the procedure linkage and function calling sequence as defined in Chapter 8.4 of the Itanium™ Software Conventions and Runtime Guide.

7.2.1 Registers

The CPU general and other registers are as defined in the Itanium™ Architecture Software Developer's Manual Volume 1 Section 3.1.

7.2.2 Floating Point Registers

The floating point registers are as defined in the Itanium™ Architecture Software Developer's Manual Volume 1 Section 3.1.

7.2.3 Stack Frame

The stackframe layout is as described in the Itanium™ Software Conventions and Runtime Guide Chapter 8.4.

7.2.4 Arguments

7.2.4.1 Introduction

The procedure parameter passing mechanism is as described in the Itanium™ Software Conventions and Runtime Guide Chapter 8.5. The following subsections provide additional information.

7.2.4.2 Integral/Pointer

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

7.2.4.3 Floating Point

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

7.2.4.4 Struct and Union Point

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

7.2.4.5 Variable Arguments

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.4.

7.2.5 Return Values

7.2.5.1 Introduction

Values are returned from functions as described in Itanium™ Software Conventions and Runtime Guide Chapter 8.6, and as further described here.

7.2.5.2 Void

Functions that return no value (void functions) are not required to put any particular value in any general register.

7.2.5.3 Integral/Pointer

See Itanium™ Software Conventions and Runtime Guide Chapter 8.6.

7.2.5.4 Floating Point

See Itanium™ Software Conventions and Runtime Guide Chapter 8.6.

7.2.5.5 Struct and Union

See Itanium™ Software Conventions and Runtime Guide Chapter 8.6 (aggregate return values). Depending on the size (including any padding), aggregate data types may be passed in one or more general registers, or in memory.

7.3 Operating System Interface

LSB-conforming applications shall use the Operating System Interfaces as defined in Chapter 3 of the Intel® Itanium™ Processor-specific Application Binary Interface.

7.3.1 Processor Execution Mode

Applications must assume that they will execute in the least privileged user mode (i.e. level 3). Other privilege levels are reserved for the Operating System.

7.3.2 Exception Interface

7.3.2.1 Introduction

LSB-conforming implementations shall support the exception interface as specified in Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.1.

7.3.2.2 Hardware Exception Types

See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.1.

7.3.2.3 Software Trap Types

See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.1.

7.3.3 Signal Delivery

LSB-conforming systems shall deliver signals as specified in Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.2.

7.3.3.1 Signal Handler Interface

The signal handler interface shall be as specified in Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.3.

7.3.4 Debugging Support

The LSB does not specify debugging information.

7.3.5 Process Startup

LSB-conforming systems shall initialize processes as specified in Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.5.

7.4 Process Initialization

LSB-conforming applications shall use the Process Startup as defined in Section 3.3.5 of the Intel® Itanium™ Processor-specific Application Binary Interface.

7.4.1 Special Registers

Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.5, defines required register initializations for process startup.

7.4.2 Process Stack (on entry)

As defined in Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.5, the return pointer register (rp) shall contain a valid return address, such that if the application program returns from the main entry routine, the implementation shall cause the application to exit normally, using the returned value as the exit status. Further, the unwind information for this "bottom of stack" routine in the implementation shall provide a mechanism for recognizing the bottom of the stack during a stack unwind.

7.4.3 Auxiliary Vector

The auxiliary vector conveys information from the operating system to the application. Only the terminating null auxiliary vector entry is required, but if any other entries are present, they shall be interpreted as follows. This vector is an array of the following structures.

```
typedef struct
{
    long int a_type;           /* Entry type */
    union
    {
        long int a_val;       /* Integer value */
        void *a_ptr;          /* Pointer value */
        void (*a_fcn) (void); /* Function pointer value */
    } a_un;
} auxv_t;
```

The application shall interpret the `a_un` value according to the `a_type`. Other auxiliary vector types are reserved.

The `a_type` field shall contain one of the following values:

`AT_NULL`

The last entry in the array has type `AT_NULL`. The value in `a_un` is undefined.

`AT_IGNORE`

The value in `a_un` is undefined, and should be ignored.

`AT_EXECD`

File descriptor of program

`AT_PHDR`

Program headers for program

`AT_PHEMT`

Size of program header entry

`AT_PHNUM`

Number of program headers

`AT_PAGESZ`

System page size

`AT_BASE`

Base address of interpreter

`AT_FLAGS`

Flags

`AT_ENTRY`

Entry point of program

`AT_NOTELF`

Program is not ELF

`AT_UID`

Real uid

`AT_EUID`

Effective uid

`AT_GID`

Real gid

`AT_EGID`

Effective gid

IECNORM.COM. Click to view the full PDF of ISO/IEC 23360-3-2:2021

AT_CLKTCK

Frequency of times()

AT_PLATFORM

String identifying platform.

AT_HWCAP

Machine dependent hints about processor capabilities.

AT_FPUCW

Used FPU control word

AT_DCACHEBSIZE

Data cache block size

AT_ICACHEBSIZE

Instruction cache block size

AT_UCACHEBSIZE

Unified cache block size

Note: The auxiliary vector is intended for passing information from the operating system to the program interpreter.

7.4.4 Environment

Although a pointer to the environment vector should be available as a third argument to the `main()` entry point, conforming applications should use `getenv()` to access the environment. (See POSIX 1003.1-2008 (ISO/IEC 9945-2009), Section `exec()`).

7.5 Coding Examples

7.5.1 Introduction

LSB-conforming applications may implement fundamental operations using the Coding Examples as shown below.

Sample code sequences and coding conventions can be found in Itanium™ Software Conventions and Runtime Guide, Chapter 9.

7.5.2 Code Model Overview/Architecture Constraints

As defined in Intel® Itanium™ Processor-specific Application Binary Interface, relocatable files, executable files, and shared object files that are supplied as part of an application shall use Position Independent Code, as described in Itanium™ Software Conventions and Runtime Guide, Chapter 12.

7.5.3 Position-Independent Function Prologue

See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.

7.5.4 Data Objects

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4, and Itanium™ Software Conventions and Runtime Guide, Chapter 12.3.

7.5.4.1 Absolute Load & Store

Conforming applications shall not use absolute addressing.

7.5.4.2 Position Relative Load & Store

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4.

7.5.5 Function Calls

See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.

Four types of procedure call are defined in Itanium™ Software Conventions and Runtime Guide, Chapter 8.3. Although special calling conventions are permitted, provided that the compiler and runtime library agree on these conventions, none are defined for this standard. Consequently, no application shall depend on a type of procedure call other than Direct Calls, Direct Dynamically Linked Calls, or Indirect Calls, as defined in Itanium™ Software Conventions and Runtime Guide, Chapter 8.3.

7.5.5.1 Absolute Direct Function Call

Conforming applications shall not use absolute addressing.

7.5.5.2 Absolute Indirect Function Call

Conforming applications shall not use absolute addressing.

7.5.5.3 Position-Independent Direct Function Call

See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.1.

7.5.5.4 Position-Independent Indirect Function Call

See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.2.

7.5.6 Branching

Branching is described in Itanium™ Architecture Software Developer's Manual Volume 4, Chapter 4.5.

7.5.6.1 Branch Instruction

See Itanium™ Architecture Software Developer's Manual Volume 4, Chapter 4.5.

7.5.6.2 Absolute switch() code

Conforming applications shall not use absolute addressing.

7.5.6.3 Position-Independent switch() code

Where there are several possible targets for a branch, the compiler may use a number of different code generation strategies. See Itanium™ Software Conventions and Runtime Guide, Chapter 9.1.7.

7.6 C Stack Frame**7.6.1 Variable Argument List**

See Itanium™ Software Conventions and Runtime Guide, Chapter 8.5.2, and 8.5.4.

7.6.2 Dynamic Allocation of Stack Space

The C library `alloca()` function should be used to dynamically allocate stack space.

7.7 Debug Information

The LSB does not currently specify the format of Debug information.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

8 Object Format

8.1 Introduction

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
- Intel® Itanium™ Processor-specific Application Binary Interface
- LSB Core - Generic
- this document

8.2 ELF Header

8.2.1 Machine Information

LSB-conforming applications shall use the Machine Information as defined in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4. Implementations shall support the LP64 model. It is unspecified whether or not the ILP32 model shall also be supported.

8.2.1.1 File Class

For LP64 relocatable objects, the file class value in `e_ident[EI_CLASS]` may be either `ELFCLASS32` or `ELFCLASS64`, and a conforming linker must be able to process either or both classes.

8.2.1.2 Data Encoding

Implementations shall support 2's complement, little endian data encoding. The data encoding value in `e_ident[EI_DATA]` shall contain the value `ELFDATA2LSB`.

8.2.1.3 OS Identification

The OS Identification field `e_ident[EI_OSABI]` shall contain the value `ELFOSABI_NONE`.

8.2.1.4 Processor Identification

The processor identification value held in `e_machine` shall contain the value `EM_IA_64`.

8.2.1.5 Processor Specific Flags

The flags field `e_flags` shall be as described in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.1.1.6.

The following additional processor-specific flags are defined:

Table 8-1 Additional Processor-Specific Flags

Name	Value
<code>EF_IA_64_LINUX_EXECUTABLE_STACK</code>	0x00000001

EF_IA_64_LINUX_EXECUTABLE_STACK

The stack and heap sections are executable. If this flag is not set, code can not be executed from the stack or heap.

8.3 Sections

The Itanium™ architecture defines two processor-specific section types, as described in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.

8.3.1 Special Sections

The following sections are defined in the Intel® Itanium™ Processor-specific Application Binary Interface.

Table 8-2 ELF Special Sections

Name	Type	Attributes
.got	SHT_PROGBITS	SHF_AL- LOC+SHF_WRITE+SHF_IA_64_SHORT
.IA_64.archext	SHT_IA_64_EXT	0
.IA_64.pltoff	SHT_PROGBITS	SHF_AL- LOC+SHF_WRITE+SHF_IA_64_SHORT
.IA_64.unwind	SHT_IA_64_UNWIND	SHF_AL- LOC+SHF_LINK_ORDER
.IA_64.unwind_info	SHT_PROGBITS	SHF_ALLOC
.plt	SHT_PROGBITS	SHF_ALLOC+SHF_EXECINSTR
.sbss	SHT_NOBITS	SHF_AL- LOC+SHF_WRITE+SHF_IA_64_SHORT
.sdata	SHT_PROGBITS	SHF_AL- LOC+SHF_WRITE+SHF_IA_64_SHORT
.sdata1	SHT_PROGBITS	SHF_AL- LOC+SHF_WRITE+SHF_IA_64_SHORT

.got

This section holds the Global Offset Table. See 'Coding Examples' in Chapter 3, 'Special Sections' in Chapter 4, and 'Global Offset Table' in Chapter 5 of the processor supplement for more information.

.IA_64.archext

This section holds product-specific extension bits. The link editor will perform a logical "or" of the extension bits of each object when creating an

executable so that it creates only a single `.IA_64.archext` section in the executable.

`.IA_64.pltoff`

This section holds local function descriptor entries.

`.IA_64.unwind`

This section holds the unwind function table. The contents are described in the Intel (r) Itanium (tm) Processor Specific ABI.

`.IA_64.unwind_info`

This section holds stack unwind and exception handling information. The exception handling information is programming language specific, and is unspecified.

`.plt`

This section holds the procedure linkage table.

`.sbss`

This section holds uninitialized data that contribute to the program's memory image. Data objects contained in this section are recommended to be eight bytes or less in size. The system initializes the data with zeroes when the program begins to run. The section occupies no file space, as indicated by the section type `SHT_NOBITS`. The `.sbss` section is placed so it may be accessed using short direct addressing (22 bit offset from `gp`).

`.sdata`

This section and the `.sdata1` section hold initialized data that contribute to the program's memory image. Data objects contained in this section are recommended to be eight bytes or less in size. The `.sdata` and `.sdata1` sections are placed so they may be accessed using short direct addressing (22 bit offset from `gp`).

`.sdata1`

See `.sdata`.

8.3.2 Linux Special Sections

The following Linux IA-64 specific sections are defined here.

Table 8-3 Additional Special Sections

Name	Type	Attributes
<code>.opd</code>	<code>SHT_PROGBITS</code>	<code>SHF_ALLOC</code>
<code>.rela.dyn</code>	<code>SHT_REL</code>	<code>SHF_ALLOC</code>
<code>.rela.IA_64.pltoff</code>	<code>SHT_REL</code>	<code>SHF_ALLOC</code>

`.opd`

This section holds function descriptors.

`.rel.dyn`

This section holds RELA type relocation information for all sections of a shared library except the PLT.

`.rel.IA_64.pltoff`

This section holds relocation information, as described in 'Relocation' section in Chapter 4 of System V ABI Update. These relocations are applied to the `.IA_64.pltoff` section.

8.3.3 Section Types

Section Types are described in the Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.2. LSB conforming implementations are not required to use any sections in the range from `SHT_IA_64_LOPSREG` to `SHT_IA_64_HIPREG`. Additionally, LSB conforming implementations are not required to support the `SHT_IA_64_PRIORITY_INIT` section, beyond the gABI requirements for the handling of unrecognized section types, linking them into a contiguous section in the object file created by the static linker.

8.3.4 Section Attribute Flags

LSB-conforming implementations shall support the section attribute flags specified in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.2.2.

8.3.5 Special Section Types

The special section types `SHT_IA64_EXT` and `SHT_IA64_UNWIND` are defined in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.2.1.

8.4 Symbol Table

If an executable file contains a reference to a function defined in one of its associated shared objects, the symbol table section for that file shall contain an entry for that symbol. The `st_shndx` member of that symbol table entry contains `SHN_UNDEF`. This signals to the dynamic linker that the symbol definition for that function is not contained in the executable file itself. If that symbol has been allocated a procedure linkage table entry in the executable file, and the `st_value` member for that symbol table entry is non-zero, the value shall contain the virtual address of the first instruction of that procedure linkage table entry. Otherwise, the `st_value` member contains zero. This procedure linkage table entry address is used by the dynamic linker in resolving references to the address of the function.

8.5 Relocation

8.5.1 Relocation Types

LSB-conforming systems shall support the relocation types described in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.3.

9 Program Loading and Dynamic Linking

9.1 Introduction

LSB-conforming implementations shall support the object file information and system actions that create running programs as specified in the System V ABI, Intel® Itanium™ Processor-specific Application Binary Interface and as supplemented by the Linux Standard Base Specification and this document.

9.2 Program Header

The program header shall be as defined in the Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.

9.2.1 Types

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.1.

9.2.2 Flags

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.1.

9.3 Program Loading

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.2.

9.4 Dynamic Linking

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.

9.4.1 Dynamic Entries

9.4.1.1 ELF Dynamic Entries

The following dynamic entries are defined in the Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.2.

DT_PLTGOT

This entry's `d_ptr` member gives the address of the first byte in the procedure linkage table

9.4.1.2 Additional Dynamic Entries

The following dynamic entries are defined here.

DT_RELACOUNT

The number of relative relocations in `.rela.dyn`

9.4.2 Global Offset Table

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4.

9.4.3 Shared Object Dependencies

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.3.

9.4.4 Function Addresses

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.5.

9.4.5 Procedure Linkage Table

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.6.

9.4.6 Initialization and Termination Functions

See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.7.

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

III Base Libraries

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

10 Libraries

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

Only interfaces and interface details which are unique to the Itanium™ platform are defined here. This section should be used in conjunction with the corresponding section of LSB Core - Generic.

10.1 Program Interpreter/Dynamic Linker

The Program Interpreter shall be `/lib/ld-lsb-ia64.so.3`.

10.2 Interfaces for libc

Table 10-1 defines the library name and shared object name for the libc library

Table 10-1 libc Definition

Library:	libc
SONAME:	libc.so.6.1

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

[LFS] Large File Support
 [LSB] LSB Core - Generic
 [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

10.2.1 RPC

10.2.1.1 Interfaces for RPC

An LSB conforming implementation shall provide the architecture specific functions for RPC specified in Table 10-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-2 libc - RPC Function Interfaces

authnone_create(GLIBC_2.2) [SVID.4]	callrpc(GLIBC_2.2) [RPC + XDR]	clnt_create(GLIBC_2.2) [SVID.4]	clnt_pcreateerror(GLIBC_2.2) [SVID.4]
clnt_pererrno(GLIBC_2.2) [SVID.4]	clnt_perror(GLIBC_2.2) [SVID.4]	clnt_spcreateerror(GLIBC_2.2) [SVID.4]	clnt_sperrno(GLIBC_2.2) [SVID.4]
clnt_sperror(GLIBC_2.2) [SVID.4]	clntraw_create(GLIBC_2.2) [RPC + XDR]	clnttcp_create(GLIBC_2.2) [RPC + XDR]	clntudp_bufcreate(GLIBC_2.2) [RPC + XDR]

clntudp_create(GLIBC_2.2) [RPC + XDR]	key_decryptsessi on(GLIBC_2.2) [SVID.4]	pmap_getport(GLIBC_2.2) [LSB]	pmap_set(GLIBC_2.2) [LSB]
pmap_unset(GLIBC_2.2) [LSB]	svc_getreqset(GLIBC_2.2) [SVID.4]	svc_register(GLIBC_2.2) [LSB]	svc_run(GLIBC_2.2) [LSB]
svc_sendreply(GLIBC_2.2) [LSB]	svcerr_auth(GLIBC_2.2) [SVID.4]	svcerr_decode(GLIBC_2.2) [SVID.4]	svcerr_noproc(GLIBC_2.2) [SVID.4]
svcerr_noprog(GLIBC_2.2) [SVID.4]	svcerr_progvers(GLIBC_2.2) [SVID.4]	svcerr_systemerr(GLIBC_2.2) [SVID.4]	svcerr_weakauth(GLIBC_2.2) [SVID.4]
svcfld_create(GLIBC_2.2) [RPC + XDR]	svcraw_create(GLIBC_2.2) [RPC + XDR]	svctcp_create(GLIBC_2.2) [LSB]	svcudp_create(GLIBC_2.2) [LSB]
xdr_accepted_reply(GLIBC_2.2) [SVID.4]	xdr_array(GLIBC_2.2) [SVID.4]	xdr_bool(GLIBC_2.2) [SVID.4]	xdr_bytes(GLIBC_2.2) [SVID.4]
xdr_callhdr(GLIBC_2.2) [SVID.4]	xdr_callmsg(GLIBC_2.2) [SVID.4]	xdr_char(GLIBC_2.2) [SVID.4]	xdr_double(GLIBC_2.2) [SVID.4]
xdr_enum(GLIBC_2.2) [SVID.4]	xdr_float(GLIBC_2.2) [SVID.4]	xdr_free(GLIBC_2.2) [SVID.4]	xdr_int(GLIBC_2.2) [SVID.4]
xdr_long(GLIBC_2.2) [SVID.4]	xdr_opaque(GLIBC_2.2) [SVID.4]	xdr_opaque_auth(GLIBC_2.2) [SVID.4]	xdr_pointer(GLIBC_2.2) [SVID.4]
xdr_reference(GLIBC_2.2) [SVID.4]	xdr_rejected_reply(GLIBC_2.2) [SVID.4]	xdr_replymsg(GLIBC_2.2) [SVID.4]	xdr_short(GLIBC_2.2) [SVID.4]
xdr_string(GLIBC_2.2) [SVID.4]	xdr_u_char(GLIBC_2.2) [SVID.4]	xdr_u_int(GLIBC_2.2) [LSB]	xdr_u_long(GLIBC_2.2) [SVID.4]
xdr_u_short(GLIBC_2.2) [SVID.4]	xdr_union(GLIBC_2.2) [SVID.4]	xdr_vector(GLIBC_2.2) [SVID.4]	xdr_void(GLIBC_2.2) [SVID.4]
xdr_wrapstring(GLIBC_2.2) [SVID.4]	xdrmem_create(GLIBC_2.2) [SVID.4]	xdrrec_create(GLIBC_2.2) [SVID.4]	xdrrec_endofrecord(GLIBC_2.2) [RPC + XDR]
xdrrec_eof(GLIBC_2.2) [SVID.4]	xdrrec_skiprecord(GLIBC_2.2) [RPC + XDR]	xdrstdio_create(GLIBC_2.2) [LSB]	

An LSB conforming implementation shall provide the architecture specific deprecated functions for RPC specified in Table 10-3, 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 10-3 libc - RPC Deprecated Function Interfaces

key_decryptsession(GLIBC_2.2) [SVID.4]			
--	--	--	--

10.2.2 Epoll

10.2.2.1 Interfaces for Epoll

No external functions are defined for libc - Epoll in this part of the specification. See also the generic specification.

10.2.3 System Calls

10.2.3.1 Interfaces for System Calls

An LSB conforming implementation shall provide the architecture specific functions for System Calls specified in Table 10-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-4 libc - System Calls Function Interfaces

__fxstat(GLIBC_2.2) [LSB]	__getpgid(GLIBC_2.2) [LSB]	__lxstat(GLIBC_2.2) [LSB]	__xmknod(GLIBC_2.2) [LSB]
__xstat(GLIBC_2.2) [LSB]	access(GLIBC_2.2) [SUSv4]	acct(GLIBC_2.2) [LSB]	alarm(GLIBC_2.2) [SUSv4]
backtrace(GLIBC_2.2) [LSB]	backtrace_symbols(GLIBC_2.2) [LSB]	backtrace_symbols_fd(GLIBC_2.2) [LSB]	brk(GLIBC_2.2) [SUSv2]
chdir(GLIBC_2.2) [SUSv4]	chmod(GLIBC_2.2) [SUSv4]	chown(GLIBC_2.2) [SUSv4]	chroot(GLIBC_2.2) [SUSv2]
clock(GLIBC_2.2) [SUSv4]	close(GLIBC_2.2) [SUSv4]	closedir(GLIBC_2.2) [SUSv4]	creat(GLIBC_2.2) [SUSv4]
dup(GLIBC_2.2) [SUSv4]	dup2(GLIBC_2.2) [SUSv4]	execl(GLIBC_2.2) [SUSv4]	execle(GLIBC_2.2) [SUSv4]
execp(GLIBC_2.2) [SUSv4]	execv(GLIBC_2.2) [SUSv4]	execve(GLIBC_2.2) [SUSv4]	execvp(GLIBC_2.2) [SUSv4]
exit(GLIBC_2.2) [SUSv4]	fchdir(GLIBC_2.2) [SUSv4]	fchmod(GLIBC_2.2) [SUSv4]	fchown(GLIBC_2.2) [SUSv4]
fcntl(GLIBC_2.2) [LSB]	fdatasync(GLIBC_2.2) [SUSv4]	fexecve(GLIBC_2.2) [SUSv4]	flock(GLIBC_2.2) [LSB]
fork(GLIBC_2.2) [SUSv4]	fstatfs(GLIBC_2.2) [LSB]	fstatvfs(GLIBC_2.2) [SUSv4]	fsync(GLIBC_2.2) [SUSv4]
ftime(GLIBC_2.2) [SUSv3]	ftruncate(GLIBC_2.2) [SUSv4]	getcontext(GLIBC_2.2) [SUSv3]	getdtablesize(GLIBC_2.2) [LSB]
getegid(GLIBC_2.2) [SUSv4]	geteuid(GLIBC_2.2) [SUSv4]	getgid(GLIBC_2.2) [SUSv4]	getgroups(GLIBC_2.2) [SUSv4]

getitimer(GLIBC_2.2) [SUSv4]	getloadavg(GLIBC_2.2) [LSB]	getpagesize(GLIBC_2.2) [LSB]	getpgid(GLIBC_2.2) [SUSv4]
getpgrp(GLIBC_2.2) [SUSv4]	getpid(GLIBC_2.2) [SUSv4]	getppid(GLIBC_2.2) [SUSv4]	getpriority(GLIBC_2.2) [SUSv4]
getrlimit(GLIBC_2.2) [LSB]	getrusage(GLIBC_2.2) [SUSv4]	getsid(GLIBC_2.2) [SUSv4]	getuid(GLIBC_2.2) [SUSv4]
getwd(GLIBC_2.2) [SUSv3]	initgroups(GLIBC_2.2) [LSB]	ioctl(GLIBC_2.2) [LSB]	ioperm(GLIBC_2.2) [LSB]
iopl(GLIBC_2.2) [LSB]	kill(GLIBC_2.2) [LSB]	killpg(GLIBC_2.2) [SUSv4]	lchown(GLIBC_2.2) [SUSv4]
link(GLIBC_2.2) [LSB]	lockf(GLIBC_2.2) [SUSv4]	lseek(GLIBC_2.2) [SUSv4]	mkdir(GLIBC_2.2) [SUSv4]
mkfifo(GLIBC_2.2) [SUSv4]	mlock(GLIBC_2.2) [SUSv4]	mlockall(GLIBC_2.2) [SUSv4]	mmap(GLIBC_2.2) [SUSv4]
mprotect(GLIBC_2.2) [SUSv4]	mremap(GLIBC_2.2) [LSB]	msync(GLIBC_2.2) [SUSv4]	munlock(GLIBC_2.2) [SUSv4]
munlockall(GLIBC_2.2) [SUSv4]	munmap(GLIBC_2.2) [SUSv4]	nanosleep(GLIBC_2.2) [SUSv4]	nice(GLIBC_2.2) [SUSv4]
open(GLIBC_2.2) [SUSv4]	opendir(GLIBC_2.2) [SUSv4]	pathconf(GLIBC_2.2) [SUSv4]	pause(GLIBC_2.2) [SUSv4]
pipe(GLIBC_2.2) [SUSv4]	poll(GLIBC_2.2) [SUSv4]	pread(GLIBC_2.2) [SUSv4]	pselect(GLIBC_2.2) [SUSv4]
ptrace(GLIBC_2.2) [LSB]	pwrite(GLIBC_2.2) [SUSv4]	read(GLIBC_2.2) [SUSv4]	readdir(GLIBC_2.2) [SUSv4]
readdir_r(GLIBC_2.2) [SUSv4]	readlink(GLIBC_2.2) [SUSv4]	readv(GLIBC_2.2) [SUSv4]	rename(GLIBC_2.2) [SUSv4]
rmdir(GLIBC_2.2) [SUSv4]	sbrk(GLIBC_2.2) [SUSv2]	sched_get_priority_max(GLIBC_2.2) [SUSv4]	sched_get_priority_min(GLIBC_2.2) [SUSv4]
sched_getparam(GLIBC_2.2) [SUSv4]	sched_getscheduler(GLIBC_2.2) [SUSv4]	sched_rr_get_interval(GLIBC_2.2) [SUSv4]	sched_setparam(GLIBC_2.2) [SUSv4]
sched_setscheduler(GLIBC_2.2) [LSB]	sched_yield(GLIBC_2.2) [SUSv4]	select(GLIBC_2.2) [SUSv4]	setcontext(GLIBC_2.2) [SUSv3]
setegid(GLIBC_2.2) [SUSv4]	seteuid(GLIBC_2.2) [SUSv4]	setgid(GLIBC_2.2) [SUSv4]	setitimer(GLIBC_2.2) [SUSv4]
setpgid(GLIBC_2.2) [SUSv4]	setpgrp(GLIBC_2.2) [SUSv4]	setpriority(GLIBC_2.2) [SUSv4]	setregid(GLIBC_2.2) [SUSv4]
setreuid(GLIBC_2.2) [SUSv4]	setrlimit(GLIBC_2.2) [LSB]	setrlimit64(GLIBC_2.2) [LFS]	setsid(GLIBC_2.2) [SUSv4]
setuid(GLIBC_2.2) [SUSv4]	sleep(GLIBC_2.2) [SUSv4]	statfs(GLIBC_2.2) [LSB]	statvfs(GLIBC_2.2) [SUSv4]

stime(GLIBC_2.2) [LSB]	symlink(GLIBC_2.2) [SUSv4]	sync(GLIBC_2.2) [SUSv4]	sysconf(GLIBC_2.2) [LSB]
sysinfo(GLIBC_2.2) [LSB]	time(GLIBC_2.2) [SUSv4]	times(GLIBC_2.2) [SUSv4]	truncate(GLIBC_2.2) [SUSv4]
ulimit(GLIBC_2.2) [SUSv4]	umask(GLIBC_2.2) [SUSv4]	uname(GLIBC_2.2) [SUSv4]	unlink(GLIBC_2.2) [LSB]
utime(GLIBC_2.2) [SUSv4]	utimes(GLIBC_2.2) [SUSv4]	vfork(GLIBC_2.2) [SUSv3]	wait(GLIBC_2.2) [SUSv4]
wait4(GLIBC_2.2) [LSB]	waitid(GLIBC_2.2) [SUSv4]	waitpid(GLIBC_2.2) [SUSv4]	write(GLIBC_2.2) [SUSv4]
writew(GLIBC_2.2) [SUSv4]			

An LSB conforming implementation shall provide the architecture specific deprecated functions for System Calls specified in Table 10-5 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 10-5 libc - System Calls Deprecated Function Interfaces

fstatfs(GLIBC_2.2) [LSB]	getdtablesize(GLIBC_2.2) [LSB]	getpagesize(GLIBC_2.2) [LSB]	getwd(GLIBC_2.2) [SUSv3]
statfs(GLIBC_2.2) [LSB]			

10.2.4 Standard I/O

10.2.4.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the architecture specific functions for Standard I/O specified in Table 10-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-6 libc - Standard I/O Function Interfaces

_IO_feof(GLIBC_2.2) [LSB]	_IO_getc(GLIBC_2.2) [LSB]	_IO_putc(GLIBC_2.2) [LSB]	_IO_puts(GLIBC_2.2) [LSB]
__fprintf_chk(GLIBC_2.3.4) [LSB]	__printf_chk(GLIBC_2.3.4) [LSB]	__snprintf_chk(GLIBC_2.3.4) [LSB]	__sprintf_chk(GLIBC_2.3.4) [LSB]
__vfprintf_chk(GLIBC_2.3.4) [LSB]	__vprintf_chk(GLIBC_2.3.4) [LSB]	__vsnprintf_chk(GLIBC_2.3.4) [LSB]	__vsprintf_chk(GLIBC_2.3.4) [LSB]
asprintf(GLIBC_2.2) [LSB]	clearerr(GLIBC_2.2) [SUSv4]	clearerr_unlocked(GLIBC_2.2) [LSB]	ctermid(GLIBC_2.2) [SUSv4]
dprintf(GLIBC_2.2) [SUSv4]	fclose(GLIBC_2.2) [SUSv4]	fdopen(GLIBC_2.2) [SUSv4]	feof(GLIBC_2.2) [SUSv4]

feof_unlocked(GLIBC_2.2) [LSB]	ferror(GLIBC_2.2) [SUSv4]	ferror_unlocked(GLIBC_2.2) [LSB]	fflush(GLIBC_2.2) [SUSv4]
fflush_unlocked(GLIBC_2.2) [LSB]	fgetc(GLIBC_2.2) [SUSv4]	fgetc_unlocked(GLIBC_2.2) [LSB]	fgetpos(GLIBC_2.2) [SUSv4]
fgets(GLIBC_2.2) [SUSv4]	fgets_unlocked(GLIBC_2.2) [LSB]	fgetwc_unlocked(GLIBC_2.2) [LSB]	fgetws_unlocked(GLIBC_2.2) [LSB]
fileno(GLIBC_2.2) [SUSv4]	fileno_unlocked(GLIBC_2.2) [LSB]	flockfile(GLIBC_2.2) [SUSv4]	fopen(GLIBC_2.2) [SUSv4]
fprintf(GLIBC_2.2) [SUSv4]	fputc(GLIBC_2.2) [SUSv4]	fputc_unlocked(GLIBC_2.2) [LSB]	fputs(GLIBC_2.2) [SUSv4]
fputs_unlocked(GLIBC_2.2) [LSB]	fputwc_unlocked(GLIBC_2.2) [LSB]	fputws_unlocked(GLIBC_2.2) [LSB]	fread(GLIBC_2.2) [SUSv4]
fread_unlocked(GLIBC_2.2) [LSB]	freopen(GLIBC_2.2) [SUSv4]	fscanf(GLIBC_2.2) [LSB]	fseek(GLIBC_2.2) [SUSv4]
fseeko(GLIBC_2.2) [SUSv4]	fsetpos(GLIBC_2.2) [SUSv4]	ftell(GLIBC_2.2) [SUSv4]	ftello(GLIBC_2.2) [SUSv4]
fwrite(GLIBC_2.2) [SUSv4]	fwrite_unlocked(GLIBC_2.2) [LSB]	getc(GLIBC_2.2) [SUSv4]	getc_unlocked(GLIBC_2.2) [SUSv4]
getchar(GLIBC_2.2) [SUSv4]	getchar_unlocked(GLIBC_2.2) [SUSv4]	getdelim(GLIBC_2.2) [SUSv4]	getline(GLIBC_2.2) [SUSv4]
getw(GLIBC_2.2) [SUSv2]	getwc_unlocked(GLIBC_2.2) [LSB]	getwchar_unlocked(GLIBC_2.2) [LSB]	pclose(GLIBC_2.2) [SUSv4]
popen(GLIBC_2.2) [SUSv4]	printf(GLIBC_2.2) [SUSv4]	putc(GLIBC_2.2) [SUSv4]	putc_unlocked(GLIBC_2.2) [SUSv4]
putchar(GLIBC_2.2) [SUSv4]	putchar_unlocked(GLIBC_2.2) [SUSv4]	puts(GLIBC_2.2) [SUSv4]	putw(GLIBC_2.2) [SUSv2]
putwc_unlocked(GLIBC_2.2) [LSB]	putwchar_unlocked(GLIBC_2.2) [LSB]	remove(GLIBC_2.2) [SUSv4]	rewind(GLIBC_2.2) [SUSv4]
rewinddir(GLIBC_2.2) [SUSv4]	scanf(GLIBC_2.2) [LSB]	seekdir(GLIBC_2.2) [SUSv4]	setbuf(GLIBC_2.2) [SUSv4]
setbuffer(GLIBC_2.2) [LSB]	setvbuf(GLIBC_2.2) [SUSv4]	snprintf(GLIBC_2.2) [SUSv4]	sprintf(GLIBC_2.2) [SUSv4]
sscanf(GLIBC_2.2) [LSB]	telldir(GLIBC_2.2) [SUSv4]	tempnam(GLIBC_2.2) [SUSv4]	ungetc(GLIBC_2.2) [SUSv4]
vasprintf(GLIBC_2.2) [LSB]	vdprintf(GLIBC_2.2) [SUSv4]	vfprintf(GLIBC_2.2) [SUSv4]	vprintf(GLIBC_2.2) [SUSv4]

IECNORM.COM Click to view the full PDF of ISO/IEC 23360-3-2:2021

vsnprintf(GLIBC_2.2) [SUSv4]	vsprintf(GLIBC_2.2) [SUSv4]		
------------------------------	-----------------------------	--	--

An LSB conforming implementation shall provide the architecture specific deprecated functions for Standard I/O specified in Table 10-7, 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 10-7 libc - Standard I/O Deprecated Function Interfaces

tempnam(GLIBC_2.2) [SUSv4]			
----------------------------	--	--	--

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard I/O specified in Table 10-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-8 libc - Standard I/O Data Interfaces

stderr(GLIBC_2.2) [SUSv4]	stdin(GLIBC_2.2) [SUSv4]	stdout(GLIBC_2.2) [SUSv4]	
---------------------------	--------------------------	---------------------------	--

10.2.5 Signal Handling

10.2.5.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the architecture specific functions for Signal Handling specified in Table 10-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-9 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.2) [LSB]	__libc_current_sigrtmin(GLIBC_2.2) [LSB]	__sigsetjmp(GLIBC_2.2) [LSB]	__sysv_signal(GLIBC_2.2) [LSB]
__xpg_sigpause(GLIBC_2.2) [LSB]	bsd_signal(GLIBC_2.2) [SUSv3]	psignal(GLIBC_2.2) [SUSv4]	raise(GLIBC_2.2) [SUSv4]
sigaction(GLIBC_2.2) [SUSv4]	sigaddset(GLIBC_2.2) [SUSv4]	sigaltstack(GLIBC_2.2) [SUSv4]	sigandset(GLIBC_2.2) [LSB]
sigdelset(GLIBC_2.2) [SUSv4]	sigemptyset(GLIBC_2.2) [SUSv4]	sigfillset(GLIBC_2.2) [SUSv4]	sighold(GLIBC_2.2) [SUSv4]
sigignore(GLIBC_2.2) [SUSv4]	siginterrupt(GLIBC_2.2) [SUSv4]	sigisemptyset(GLIBC_2.2) [LSB]	sigismember(GLIBC_2.2) [SUSv4]
siglongjmp(GLIBC_2.2) [SUSv4]	signal(GLIBC_2.2) [SUSv4]	sigorset(GLIBC_2.2) [LSB]	sigpause(GLIBC_2.2) [LSB]
sigpending(GLIBC_2.2) [SUSv4]	sigprocmask(GLIBC_2.2) [SUSv4]	sigqueue(GLIBC_2.2) [SUSv4]	sigrelse(GLIBC_2.2) [SUSv4]
sigreturn(GLIBC_2.2) [LSB]	sigset(GLIBC_2.2) [SUSv4]	sigsuspend(GLIBC_2.2) [SUSv4]	sigtimedwait(GLIBC_2.2) [SUSv4]

sigwait(GLIBC_2.2) [SUSv4]	sigwaitinfo(GLIBC_2.2) [SUSv4]		
----------------------------	--------------------------------	--	--

An LSB conforming implementation shall provide the architecture specific deprecated functions for Signal Handling specified in Table 10-10, 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 10-10 libc - Signal Handling Deprecated Function Interfaces

sigpause(GLIBC_2.2) [LSB]			
---------------------------	--	--	--

An LSB conforming implementation shall provide the architecture specific data interfaces for Signal Handling specified in Table 10-11, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-11 libc - Signal Handling Data Interfaces

_sys_siglist(GLIBC_2.3.3) [LSB]			
---------------------------------	--	--	--

10.2.6 Localization Functions

10.2.6.1 Interfaces for Localization Functions

An LSB conforming implementation shall provide the architecture specific functions for Localization Functions specified in Table 10-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-12 libc - Localization Functions Function Interfaces

bind_textdomain_codeset(GLIBC_2.2) [LSB]	bindtextdomain(GLIBC_2.2) [LSB]	catclose(GLIBC_2.2) [SUSv4]	catgets(GLIBC_2.2) [SUSv4]
catopen(GLIBC_2.2) [SUSv4]	dcgettext(GLIBC_2.2) [LSB]	dcngettext(GLIBC_2.2) [LSB]	dgettext(GLIBC_2.2) [LSB]
dngettext(GLIBC_2.2) [LSB]	gettext(GLIBC_2.2) [LSB]	iconv(GLIBC_2.2) [SUSv4]	iconv_close(GLIBC_2.2) [SUSv4]
iconv_open(GLIBC_2.2) [SUSv4]	localeconv(GLIBC_2.2) [SUSv4]	ngettext(GLIBC_2.2) [LSB]	nl_langinfo(GLIBC_2.2) [SUSv4]
setlocale(GLIBC_2.2) [SUSv4]	textdomain(GLIBC_2.2) [LSB]		

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

Table 10-13 libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr (GLIBC_2.2) [LSB]			
--	--	--	--

10.2.7 Posix Spawn Option

10.2.7.1 Interfaces for Posix Spawn Option

An LSB conforming implementation shall provide the architecture specific functions for Posix Spawn Option specified in Table 10-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-14 libc - Posix Spawn Option Function Interfaces

posix_spawn(GLIBC_2.15) [SUSv4]	posix_spawn_file_actions_addclose(GLIBC_2.2) [SUSv4]	posix_spawn_file_actions_adddup2(GLIBC_2.2) [SUSv4]	posix_spawn_file_actions_addopen(GLIBC_2.2) [SUSv4]
posix_spawn_file_actions_destroy(GLIBC_2.2) [SUSv4]	posix_spawn_file_actions_init(GLIBC_2.2) [SUSv4]	posix_spawnattr_destroy(GLIBC_2.2) [SUSv4]	posix_spawnattr_getflags(GLIBC_2.2) [SUSv4]
posix_spawnattr_getpgroup(GLIBC_2.2) [SUSv4]	posix_spawnattr_getschedparam(GLIBC_2.2) [SUSv4]	posix_spawnattr_getschedpolicy(GLIBC_2.2) [SUSv4]	posix_spawnattr_getsigdefault(GLIBC_2.2) [SUSv4]
posix_spawnattr_getsigmask(GLIBC_2.2) [SUSv4]	posix_spawnattr_init(GLIBC_2.2) [SUSv4]	posix_spawnattr_setflags(GLIBC_2.2) [SUSv4]	posix_spawnattr_setpgroup(GLIBC_2.2) [SUSv4]
posix_spawnattr_setschedparam(GLIBC_2.2) [SUSv4]	posix_spawnattr_setschedpolicy(GLIBC_2.2) [SUSv4]	posix_spawnattr_setsigdefault(GLIBC_2.2) [SUSv4]	posix_spawnattr_setsigmask(GLIBC_2.2) [SUSv4]
posix_spawnnp(GLIBC_2.15) [SUSv4]			

10.2.8 Posix Advisory Option

10.2.8.1 Interfaces for Posix Advisory Option

An LSB conforming implementation shall provide the architecture specific functions for Posix Advisory Option specified in Table 10-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-15 libc - Posix Advisory Option Function Interfaces

posix_fadvise(GLIBC_2.2) [SUSv4]	posix_fallocate(GLIBC_2.2) [SUSv4]	posix_madvise(GLIBC_2.2) [SUSv4]	posix_memalign(GLIBC_2.2) [SUSv4]
-------------------------------------	---------------------------------------	-------------------------------------	--------------------------------------

10.2.9 Socket Interface

10.2.9.1 Interfaces for Socket Interface

An LSB conforming implementation shall provide the architecture specific functions for Socket Interface specified in Table 10-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-16 libc - Socket Interface Function Interfaces

__h_errno_location(GLIBC_2.2) [LSB]	accept(GLIBC_2.2) [SUSv4]	bind(GLIBC_2.2) [SUSv4]	bindresvport(GLIBC_2.2) [LSB]
connect(GLIBC_2.2) [SUSv4]	gethostid(GLIBC_2.2) [SUSv4]	gethostname(GLIBC_2.2) [SUSv4]	getpeername(GLIBC_2.2) [SUSv4]
getsockname(GLIBC_2.2) [SUSv4]	getsockopt(GLIBC_2.2) [LSB]	if_freenameindex(GLIBC_2.2) [SUSv4]	if_indextoname(GLIBC_2.2) [SUSv4]
if_nameindex(GLIBC_2.2) [SUSv4]	if_nametoindex(GLIBC_2.2) [SUSv4]	listen(GLIBC_2.2) [SUSv4]	recv(GLIBC_2.2) [SUSv4]
recvfrom(GLIBC_2.2) [SUSv4]	recvmsg(GLIBC_2.2) [SUSv4]	send(GLIBC_2.2) [SUSv4]	sendmsg(GLIBC_2.2) [SUSv4]
sendto(GLIBC_2.2) [SUSv4]	setsockopt(GLIBC_2.2) [LSB]	shutdown(GLIBC_2.2) [SUSv4]	socketatmark(GLIBC_2.2.4) [SUSv4]
socket(GLIBC_2.2) [SUSv4]	socketpair(GLIBC_2.2) [SUSv4]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Socket Interface specified in Table 10-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-17 libc - Socket Interface Data Interfaces

in6addr_any(GLIBC_2.2) [SUSv3]	in6addr_loopback(GLIBC_2.2) [SUSv3]		
--------------------------------	-------------------------------------	--	--

10.2.10 Wide Characters

10.2.10.1 Interfaces for Wide Characters

An LSB conforming implementation shall provide the architecture specific functions for Wide Characters specified in Table 10-18, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-18 libc - Wide Characters Function Interfaces

__wcstod_internal(GLIBC_2.2) [LSB]	__wcstof_internal(GLIBC_2.2) [LSB]	__wcstol_internal(GLIBC_2.2) [LSB]	__wcstold_internal(GLIBC_2.2) [LSB]
------------------------------------	------------------------------------	------------------------------------	-------------------------------------

__wcstoul_internal(GLIBC_2.2) [LSB]	btowc(GLIBC_2.2) [SUSv4]	fgetwc(GLIBC_2.2) [SUSv4]	fgetws(GLIBC_2.2) [SUSv4]
fputwc(GLIBC_2.2) [SUSv4]	fputws(GLIBC_2.2) [SUSv4]	fwide(GLIBC_2.2) [SUSv4]	fwprintf(GLIBC_2.2) [SUSv4]
fwscanf(GLIBC_2.2) [LSB]	getwc(GLIBC_2.2) [SUSv4]	getwchar(GLIBC_2.2) [SUSv4]	mblen(GLIBC_2.2) [SUSv4]
mbrlen(GLIBC_2.2) [SUSv4]	mbrtowc(GLIBC_2.2) [SUSv4]	mbsinit(GLIBC_2.2) [SUSv4]	mbsnrtowcs(GLIBC_2.2) [SUSv4]
mbsrtowcs(GLIBC_2.2) [SUSv4]	mbstowcs(GLIBC_2.2) [SUSv4]	mbtowc(GLIBC_2.2) [SUSv4]	putwc(GLIBC_2.2) [SUSv4]
putwchar(GLIBC_2.2) [SUSv4]	swprintf(GLIBC_2.2) [SUSv4]	swscanf(GLIBC_2.2) [LSB]	towctrans(GLIBC_2.2) [SUSv4]
towlower(GLIBC_2.2) [SUSv4]	towupper(GLIBC_2.2) [SUSv4]	ungetwc(GLIBC_2.2) [SUSv4]	vfwprintf(GLIBC_2.2) [SUSv4]
vfwscanf(GLIBC_2.2) [LSB]	vswprintf(GLIBC_2.2) [SUSv4]	vswscanf(GLIBC_2.2) [LSB]	vwprintf(GLIBC_2.2) [SUSv4]
vwsscanf(GLIBC_2.2) [LSB]	wcpcpy(GLIBC_2.2) [SUSv4]	wcpncpy(GLIBC_2.2) [SUSv4]	wcrtomb(GLIBC_2.2) [SUSv4]
wscasecmp(GLIBC_2.2) [SUSv4]	wscat(GLIBC_2.2) [SUSv4]	wcschr(GLIBC_2.2) [SUSv4]	wscmp(GLIBC_2.2) [SUSv4]
wscoll(GLIBC_2.2) [SUSv4]	wcscpy(GLIBC_2.2) [SUSv4]	wcscspn(GLIBC_2.2) [SUSv4]	wcsdup(GLIBC_2.2) [SUSv4]
wcsftime(GLIBC_2.2) [SUSv4]	wcslen(GLIBC_2.2) [SUSv4]	wcsncasecmp(GLIBC_2.2) [SUSv4]	wcsncat(GLIBC_2.2) [SUSv4]
wcsncmp(GLIBC_2.2) [SUSv4]	wcsncpy(GLIBC_2.2) [SUSv4]	wcsnlen(GLIBC_2.2) [SUSv4]	wcsnrtombs(GLIBC_2.2) [SUSv4]
wcspbrk(GLIBC_2.2) [SUSv4]	wcsrchr(GLIBC_2.2) [SUSv4]	wcsrtombs(GLIBC_2.2) [SUSv4]	wcsspn(GLIBC_2.2) [SUSv4]
wcsstr(GLIBC_2.2) [SUSv4]	wctod(GLIBC_2.2) [SUSv4]	wctof(GLIBC_2.2) [SUSv4]	wctoimax(GLIBC_2.2) [SUSv4]
wcstok(GLIBC_2.2) [SUSv4]	wctol(GLIBC_2.2) [SUSv4]	wctold(GLIBC_2.2) [SUSv4]	wctoll(GLIBC_2.2) [SUSv4]
wcstombs(GLIBC_2.2) [SUSv4]	wctoq(GLIBC_2.2) [LSB]	wctoul(GLIBC_2.2) [SUSv4]	wctoull(GLIBC_2.2) [SUSv4]
wcstoumax(GLIBC_2.2) [SUSv4]	wctouq(GLIBC_2.2) [LSB]	wcswcs(GLIBC_2.2) [SUSv3]	wcswidth(GLIBC_2.2) [SUSv4]
wcsxfrm(GLIBC_2.2) [SUSv4]	wctob(GLIBC_2.2) [SUSv4]	wctomb(GLIBC_2.2) [SUSv4]	wctrans(GLIBC_2.2) [SUSv4]
wctype(GLIBC_2.2) [SUSv4]	wcwidth(GLIBC_2.2) [SUSv4]	wmemchr(GLIBC_2.2) [SUSv4]	wmemcmp(GLIBC_2.2) [SUSv4]

wmemcpy(GLIBC_2.2) [SUSv4]	wmemmove(GLIBC_2.2) [SUSv4]	wmemset(GLIBC_2.2) [SUSv4]	wprintf(GLIBC_2.2) [SUSv4]
wscanf(GLIBC_2.2) [LSB]			

10.2.11 String Functions

10.2.11.1 Interfaces for String Functions

An LSB conforming implementation shall provide the architecture specific functions for String Functions specified in Table 10-19, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-19 libc - String Functions Function Interfaces

__memcpy(GLIBC_2.2) [LSB]	__rawmemchr(GLIBC_2.2) [LSB]	__stpcpy(GLIBC_2.2) [LSB]	__strdup(GLIBC_2.2) [LSB]
__strtod_internal(GLIBC_2.2) [LSB]	__strtof_internal(GLIBC_2.2) [LSB]	__strtok_r(GLIBC_2.2) [LSB]	__strtol_internal(GLIBC_2.2) [LSB]
__strtold_internal(GLIBC_2.2) [LSB]	__strtoll_internal(GLIBC_2.2) [LSB]	__strtoul_internal(GLIBC_2.2) [LSB]	__strtoull_internal(GLIBC_2.2) [LSB]
__xpg_strerror_r(GLIBC_2.3.4) [LSB]	bcmp(GLIBC_2.2) [SUSv3]	bcopy(GLIBC_2.2) [SUSv3]	bzero(GLIBC_2.2) [SUSv3]
ffs(GLIBC_2.2) [SUSv4]	index(GLIBC_2.2) [SUSv3]	memcpy(GLIBC_2.2) [SUSv4]	memchr(GLIBC_2.2) [SUSv4]
memcmp(GLIBC_2.2) [SUSv4]	memcpy(GLIBC_2.2) [SUSv4]	memmove(GLIBC_2.2) [SUSv4]	memrchr(GLIBC_2.2) [LSB]
memset(GLIBC_2.2) [SUSv4]	rindex(GLIBC_2.2) [SUSv3]	stpcpy(GLIBC_2.2) [SUSv4]	stpncpy(GLIBC_2.2) [SUSv4]
strcasemp(GLIBC_2.2) [SUSv4]	strcasestr(GLIBC_2.2) [LSB]	strcat(GLIBC_2.2) [SUSv4]	strchr(GLIBC_2.2) [SUSv4]
strcmp(GLIBC_2.2) [SUSv4]	strcoll(GLIBC_2.2) [SUSv4]	strcpy(GLIBC_2.2) [SUSv4]	strcspn(GLIBC_2.2) [SUSv4]
strdup(GLIBC_2.2) [SUSv4]	strerror(GLIBC_2.2) [SUSv4]	strerror_r(GLIBC_2.2) [LSB]	strfmon(GLIBC_2.2) [SUSv4]
strftime(GLIBC_2.2) [SUSv4]	strlen(GLIBC_2.2) [SUSv4]	strncasemp(GLIBC_2.2) [SUSv4]	strncat(GLIBC_2.2) [SUSv4]
strncmp(GLIBC_2.2) [SUSv4]	strncpy(GLIBC_2.2) [SUSv4]	strndup(GLIBC_2.2) [SUSv4]	strnlen(GLIBC_2.2) [SUSv4]
strpbrk(GLIBC_2.2) [SUSv4]	strptime(GLIBC_2.2) [LSB]	strrchr(GLIBC_2.2) [SUSv4]	strsep(GLIBC_2.2) [LSB]
strsignal(GLIBC_2.2) [SUSv4]	strspn(GLIBC_2.2) [SUSv4]	strstr(GLIBC_2.2) [SUSv4]	strtof(GLIBC_2.2) [SUSv4]

IEC NORMATIVE DOCUMENT - Full PDF of ISO/IEC 23360-3-2:2021

strtoimax(GLIBC_2.2) [SUSv4]	strtok(GLIBC_2.2) [SUSv4]	strtok_r(GLIBC_2.2) [SUSv4]	strtold(GLIBC_2.2) [SUSv4]
strtoll(GLIBC_2.2) [SUSv4]	strtoq(GLIBC_2.2) [LSB]	strtoull(GLIBC_2.2) [SUSv4]	strtoumax(GLIBC_2.2) [SUSv4]
strtouq(GLIBC_2.2) [LSB]	strxfrm(GLIBC_2.2) [SUSv4]	swab(GLIBC_2.2) [SUSv4]	

An LSB conforming implementation shall provide the architecture specific deprecated functions for String Functions specified in Table 10-20, 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 10-20 libc - String Functions Deprecated Function Interfaces

strerror_r(GLIBC_2.2) [LSB]			
-----------------------------	--	--	--

10.2.12 IPC Functions

10.2.12.1 Interfaces for IPC Functions

An LSB conforming implementation shall provide the architecture specific functions for IPC Functions specified in Table 10-21, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-21 libc - IPC Functions Function Interfaces

ftok(GLIBC_2.2) [SUSv4]	msgctl(GLIBC_2.2) [SUSv4]	msgget(GLIBC_2.2) [SUSv4]	msgrcv(GLIBC_2.2) [SUSv4]
msgsnd(GLIBC_2.2) [SUSv4]	semctl(GLIBC_2.2) [SUSv4]	semget(GLIBC_2.2) [SUSv4]	semop(GLIBC_2.2) [SUSv4]
shmat(GLIBC_2.2) [SUSv4]	shmctl(GLIBC_2.2) [SUSv4]	shmdt(GLIBC_2.2) [SUSv4]	shmget(GLIBC_2.2) [SUSv4]

10.2.13 Regular Expressions

10.2.13.1 Interfaces for Regular Expressions

An LSB conforming implementation shall provide the architecture specific functions for Regular Expressions specified in Table 10-22, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-22 libc - Regular Expressions Function Interfaces

regcomp(GLIBC_2.2) [SUSv4]	regerror(GLIBC_2.2) [SUSv4]	regexexec(GLIBC_2.3.4) [LSB]	regfree(GLIBC_2.2) [SUSv4]
----------------------------	-----------------------------	------------------------------	----------------------------

10.2.14 Character Type Functions

10.2.14.1 Interfaces for Character Type Functions

An LSB conforming implementation shall provide the architecture specific functions for Character Type Functions specified in Table 10-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-23 libc - Character Type Functions Function Interfaces

__ctype_get_mb_cur_max(GLIBC_2.2) [LSB]	_tolower(GLIBC_2.2) [SUSv4]	_toupper(GLIBC_2.2) [SUSv4]	isalnum(GLIBC_2.2) [SUSv4]
isalpha(GLIBC_2.2) [SUSv4]	isascii(GLIBC_2.2) [SUSv4]	iscntrl(GLIBC_2.2) [SUSv4]	isdigit(GLIBC_2.2) [SUSv4]
isgraph(GLIBC_2.2) [SUSv4]	islower(GLIBC_2.2) [SUSv4]	isprint(GLIBC_2.2) [SUSv4]	ispunct(GLIBC_2.2) [SUSv4]
isspace(GLIBC_2.2) [SUSv4]	isupper(GLIBC_2.2) [SUSv4]	iswalnum(GLIBC_2.2) [SUSv4]	iswalpha(GLIBC_2.2) [SUSv4]
iswblank(GLIBC_2.2) [SUSv4]	iswcntrl(GLIBC_2.2) [SUSv4]	iswctype(GLIBC_2.2) [SUSv4]	iswdigit(GLIBC_2.2) [SUSv4]
iswgraph(GLIBC_2.2) [SUSv4]	iswlower(GLIBC_2.2) [SUSv4]	iswprint(GLIBC_2.2) [SUSv4]	iswpunct(GLIBC_2.2) [SUSv4]
iswspace(GLIBC_2.2) [SUSv4]	iswupper(GLIBC_2.2) [SUSv4]	iswxdigit(GLIBC_2.2) [SUSv4]	isxdigit(GLIBC_2.2) [SUSv4]
toascii(GLIBC_2.2) [SUSv4]	tolower(GLIBC_2.2) [SUSv4]	toupper(GLIBC_2.2) [SUSv4]	

10.2.15 Time Manipulation

10.2.15.1 Interfaces for Time Manipulation

An LSB conforming implementation shall provide the architecture specific functions for Time Manipulation specified in Table 10-24, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-24 libc - Time Manipulation Function Interfaces

adjtime(GLIBC_2.2) [LSB]	asctime(GLIBC_2.2) [SUSv4]	asctime_r(GLIBC_2.2) [SUSv4]	ctime(GLIBC_2.2) [SUSv4]
ctime_r(GLIBC_2.2) [SUSv4]	difftime(GLIBC_2.2) [SUSv4]	gmtime(GLIBC_2.2) [SUSv4]	gmtime_r(GLIBC_2.2) [SUSv4]
localtime(GLIBC_2.2) [SUSv4]	localtime_r(GLIBC_2.2) [SUSv4]	mktime(GLIBC_2.2) [SUSv4]	tzset(GLIBC_2.2) [SUSv4]
ualarm(GLIBC_2.2) [SUSv3]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Time Manipulation specified in Table 10-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-25 libc - Time Manipulation Data Interfaces

__daylight(GLIBC_2.2) [LSB]	__timezone(GLIBC_2.2) [LSB]	__tzname(GLIBC_2.2) [LSB]	daylight(GLIBC_2.2) [SUSv4]
timezone(GLIBC_2.2) [SUSv4]	tzname(GLIBC_2.2) [SUSv4]		

10.2.16 Terminal Interface Functions

10.2.16.1 Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the architecture specific functions for Terminal Interface Functions specified in Table 10-26, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-26 libc - Terminal Interface Functions Function Interfaces

cfgetispeed(GLIBC_2.2) [SUSv4]	cfgetospeed(GLIBC_2.2) [SUSv4]	cfmakeraw(GLIBC_2.2) [LSB]	cfsetispeed(GLIBC_2.2) [SUSv4]
cfsetospeed(GLIBC_2.2) [SUSv4]	cfsetspeed(GLIBC_2.2) [LSB]	tcdrain(GLIBC_2.2) [SUSv4]	tcflow(GLIBC_2.2) [SUSv4]
tcflush(GLIBC_2.2) [SUSv4]	tcgetattr(GLIBC_2.2) [SUSv4]	tcgetpgrp(GLIBC_2.2) [SUSv4]	tcgetsid(GLIBC_2.2) [SUSv4]
tcsendbreak(GLIBC_2.2) [SUSv4]	tcsetattr(GLIBC_2.2) [SUSv4]	tcsetpgrp(GLIBC_2.2) [SUSv4]	

10.2.17 System Database Interface

10.2.17.1 Interfaces for System Database Interface

An LSB conforming implementation shall provide the architecture specific functions for System Database Interface specified in Table 10-27, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-27 libc - System Database Interface Function Interfaces

endgrent(GLIBC_2.2) [SUSv4]	endprotoent(GLIBC_2.2) [SUSv4]	endpwent(GLIBC_2.2) [SUSv4]	endservent(GLIBC_2.2) [SUSv4]
endutent(GLIBC_2.2) [LSB]	endutxent(GLIBC_2.2) [SUSv4]	getgrent(GLIBC_2.2) [SUSv4]	getgrent_r(GLIBC_2.2) [LSB]
getgrgid(GLIBC_2.2) [SUSv4]	getgrgid_r(GLIBC_2.2) [SUSv4]	getgrnam(GLIBC_2.2) [SUSv4]	getgrnam_r(GLIBC_2.2) [SUSv4]
getgrouplist(GLIBC_2.2.4) [LSB]	gethostbyaddr(GLIBC_2.2) [SUSv3]	gethostbyaddr_r(GLIBC_2.2) [LSB]	gethostbyname(GLIBC_2.2) [SUSv3]
gethostbyname2(GLIBC_2.2) [LSB]	gethostbyname2_r(GLIBC_2.2) [LSB]	gethostbyname_r(GLIBC_2.2) [LSB]	getprotobyname(GLIBC_2.2) [SUSv4]
getprotobyname_r(GLIBC_2.2) [LSB]	getprotobynumber(GLIBC_2.2) [SUSv4]	getprotobynumber_r(GLIBC_2.2) [LSB]	getprotoent(GLIBC_2.2) [SUSv4]

getprotoent_r(GLIBC_2.2) [LSB]	getpwent(GLIBC_2.2) [SUSv4]	getpwent_r(GLIBC_2.2) [LSB]	getpwnam(GLIBC_2.2) [SUSv4]
getpwnam_r(GLIBC_2.2) [SUSv4]	getpwuid(GLIBC_2.2) [SUSv4]	getpwuid_r(GLIBC_2.2) [SUSv4]	getservbyname(GLIBC_2.2) [SUSv4]
getservbyname_r(GLIBC_2.2) [LSB]	getservbyport(GLIBC_2.2) [SUSv4]	getservbyport_r(GLIBC_2.2) [LSB]	getservent(GLIBC_2.2) [SUSv4]
getservent_r(GLIBC_2.2) [LSB]	getutent(GLIBC_2.2) [LSB]	getutent_r(GLIBC_2.2) [LSB]	getutxent(GLIBC_2.2) [SUSv4]
getutxid(GLIBC_2.2) [SUSv4]	getutxline(GLIBC_2.2) [SUSv4]	pututxline(GLIBC_2.2) [SUSv4]	setgrent(GLIBC_2.2) [SUSv4]
setgroups(GLIBC_2.2) [LSB]	setprotoent(GLIBC_2.2) [SUSv4]	setpwent(GLIBC_2.2) [SUSv4]	setservent(GLIBC_2.2) [SUSv4]
setutent(GLIBC_2.2) [LSB]	setutxent(GLIBC_2.2) [SUSv4]	utmpname(GLIBC_2.2) [LSB]	

An LSB conforming implementation shall provide the architecture specific deprecated functions for System Database Interface specified in Table 10-28, 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 10-28 libc - System Database Interface Deprecated Function Interfaces

gethostbyaddr(GLIBC_2.2) [SUSv3]	gethostbyaddr_r(GLIBC_2.2) [LSB]	gethostbyname(GLIBC_2.2) [SUSv3]	gethostbyname2(GLIBC_2.2) [LSB]
gethostbyname2_r(GLIBC_2.2) [LSB]	gethostbyname_r(GLIBC_2.2) [LSB]		

10.2.18 Language Support

10.2.18.1 Interfaces for Language Support

An LSB conforming implementation shall provide the architecture specific functions for Language Support specified in Table 10-29, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-29 libc - Language Support Function Interfaces

__libc_start_main(GLIBC_2.2) [LSB]			
------------------------------------	--	--	--

10.2.19 Large File Support

10.2.19.1 Interfaces for Large File Support

An LSB conforming implementation shall provide the architecture specific functions for Large File Support specified in Table 10-30, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-30 libc - Large File Support Function Interfaces

__fxstat64(GLIBC_2.2) [LSB]	__lxstat64(GLIBC_2.2) [LSB]	__xstat64(GLIBC_2.2) [LSB]	creat64(GLIBC_2.2) [LFS]
fgetpos64(GLIBC_2.2) [LFS]	fopen64(GLIBC_2.2) [LFS]	freopen64(GLIBC_2.2) [LFS]	fseeko64(GLIBC_2.2) [LFS]
fsetpos64(GLIBC_2.2) [LFS]	fstatfs64(GLIBC_2.2) [LSB]	fstatvfs64(GLIBC_2.2) [LFS]	ftello64(GLIBC_2.2) [LFS]
ftruncate64(GLIBC_2.2) [LFS]	ftw64(GLIBC_2.2) [LFS]	getrlimit64(GLIBC_2.2) [LFS]	lockf64(GLIBC_2.2) [LFS]
lseek64(GLIBC_2.2) [LFS]	mkstemp64(GLIBC_2.2) [LSB]	mmap64(GLIBC_2.2) [LFS]	nftw64(GLIBC_2.3) [LFS]
open64(GLIBC_2.2) [LFS]	posix_fadvise64(GLIBC_2.2) [LSB]	posix_fallocate64(GLIBC_2.2) [LSB]	pread64(GLIBC_2.2) [LSB]
pwrite64(GLIBC_2.2) [LSB]	readdir64(GLIBC_2.2) [LFS]	readdir64_r(GLIBC_2.2) [LSB]	statfs64(GLIBC_2.2) [LSB]
statvfs64(GLIBC_2.2) [LFS]	tmpfile64(GLIBC_2.2) [LFS]	truncate64(GLIBC_2.2) [LFS]	

An LSB conforming implementation shall provide the architecture specific deprecated functions for Large File Support specified in Table 10-31, with the full mandatory functionality as described in the referenced underlying specification.

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

Table 10-31 libc - Large File Support Deprecated Function Interfaces

fstatfs64(GLIBC_2.2) [LSB]	statfs64(GLIBC_2.2) [LSB]		
----------------------------	---------------------------	--	--

10.2.20 Inotify

10.2.20.1 Interfaces for Inotify

No external functions are defined for libc - Inotify in this part of the specification. See also the generic specification.

10.2.21 Standard Library

10.2.21.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the architecture specific functions for Standard Library specified in Table 10-32, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-32 libc - Standard Library Function Interfaces

_Exit(GLIBC_2.2) [SUSv4]	__assert_fail(GLIBC_2.2) [LSB]	__cxa_atexit(GLIBC_2.2) [LSB]	__cxa_finalize(GLIBC_2.2) [LSB]
__errno_location(GLIBC_2.2) [LSB]	__fpending(GLIBC_2.2) [LSB]	__getpagesize(GLIBC_2.2) [LSB]	__isinf(GLIBC_2.2) [LSB]
__isinf(GLIBC_2.2) [LSB]	__isinfl(GLIBC_2.2) [LSB]	__isnan(GLIBC_2.2) [LSB]	__isnanf(GLIBC_2.2) [LSB]
__isnandl(GLIBC_2.2) [LSB]	__sysconf(GLIBC_2.2) [LSB]	__xpg_basename(GLIBC_2.2) [LSB]	_exit(GLIBC_2.2) [SUSv4]
_longjmp(GLIBC_2.2) [SUSv4]	_setjmp(GLIBC_2.2) [SUSv4]	a64l(GLIBC_2.2) [SUSv4]	abort(GLIBC_2.2) [SUSv4]
abs(GLIBC_2.2) [SUSv4]	alphasort(GLIBC_2.2) [SUSv4]	alphasort64(GLIBC_2.2) [LSB]	argz_add(GLIBC_2.2) [LSB]
argz_add_sep(GLIBC_2.2) [LSB]	argz_append(GLIBC_2.2) [LSB]	argz_count(GLIBC_2.2) [LSB]	argz_create(GLIBC_2.2) [LSB]
argz_create_sep(GLIBC_2.2) [LSB]	argz_delete(GLIBC_2.2) [LSB]	argz_extract(GLIBC_2.2) [LSB]	argz_insert(GLIBC_2.2) [LSB]
argz_next(GLIBC_2.2) [LSB]	argz_replace(GLIBC_2.2) [LSB]	argz_stringify(GLIBC_2.2) [LSB]	atof(GLIBC_2.2) [SUSv4]
atoi(GLIBC_2.2) [SUSv4]	atol(GLIBC_2.2) [SUSv4]	atoll(GLIBC_2.2) [SUSv4]	basename(GLIBC_2.2) [LSB]
bsearch(GLIBC_2.2) [SUSv4]	calloc(GLIBC_2.2) [SUSv4]	closelog(GLIBC_2.2) [SUSv4]	confstr(GLIBC_2.2) [SUSv4]
cuserid(GLIBC_2.2) [SUSv2]	daemon(GLIBC_2.2) [LSB]	dirfd(GLIBC_2.2) [SUSv4]	dirname(GLIBC_2.2) [SUSv4]
div(GLIBC_2.2) [SUSv4]	dl_iterate_phdr(GLIBC_2.2.4) [LSB]	drand48(GLIBC_2.2) [SUSv4]	drand48_r(GLIBC_2.2) [LSB]
ecvt(GLIBC_2.2) [SUSv3]	envz_add(GLIBC_2.2) [LSB]	envz_entry(GLIBC_2.2) [LSB]	envz_get(GLIBC_2.2) [LSB]
envz_merge(GLIBC_2.2) [LSB]	envz_remove(GLIBC_2.2) [LSB]	envz_strip(GLIBC_2.2) [LSB]	erand48(GLIBC_2.2) [SUSv4]
erand48_r(GLIBC_2.2) [LSB]	err(GLIBC_2.2) [LSB]	error(GLIBC_2.2) [LSB]	errx(GLIBC_2.2) [LSB]
fcvt(GLIBC_2.2) [SUSv3]	fmemopen(GLIBC_2.2) [SUSv4]	fmsg(GLIBC_2.2) [SUSv4]	fnmatch(GLIBC_2.2.3) [LSB]

fpathconf(GLIBC_2.2) [SUSv4]	free(GLIBC_2.2) [SUSv4]	freeaddrinfo(GLIBC_2.2) [SUSv4]	ftwlockfile(GLIBC_2.2) [SUSv4]
ftw(GLIBC_2.2) [SUSv4]	funlockfile(GLIBC_2.2) [SUSv4]	gai_strerror(GLIBC_2.2) [SUSv4]	gcvt(GLIBC_2.2) [SUSv3]
getaddrinfo(GLIBC_2.2) [SUSv4]	getcwd(GLIBC_2.2) [LSB]	getdate(GLIBC_2.2) [SUSv4]	getdomainname(GLIBC_2.2) [LSB]
getenv(GLIBC_2.2) [SUSv4]	getlogin(GLIBC_2.2) [SUSv4]	getlogin_r(GLIBC_2.2) [SUSv4]	getnameinfo(GLIBC_2.2) [SUSv4]
getopt(GLIBC_2.2) [LSB]	getopt_long(GLIBC_2.2) [LSB]	getopt_long_only(GLIBC_2.2) [LSB]	getsubopt(GLIBC_2.2) [SUSv4]
gettimeofday(GLIBC_2.2) [SUSv4]	glob(GLIBC_2.2) [SUSv4]	glob64(GLIBC_2.2) [LSB]	globfree(GLIBC_2.2) [SUSv4]
globfree64(GLIBC_2.2) [LSB]	grantpt(GLIBC_2.2) [SUSv4]	hcreate(GLIBC_2.2) [SUSv4]	hcreate_r(GLIBC_2.2) [LSB]
hdestroy(GLIBC_2.2) [SUSv4]	hdestroy_r(GLIBC_2.2) [LSB]	hsearch(GLIBC_2.2) [SUSv4]	hsearch_r(GLIBC_2.2) [LSB]
htonl(GLIBC_2.2) [SUSv4]	htons(GLIBC_2.2) [SUSv4]	imaxabs(GLIBC_2.2) [SUSv4]	imaxdiv(GLIBC_2.2) [SUSv4]
inet_addr(GLIBC_2.2) [SUSv4]	inet_aton(GLIBC_2.2) [LSB]	inet_ntoa(GLIBC_2.2) [SUSv4]	inet_ntop(GLIBC_2.2) [SUSv4]
inet_pton(GLIBC_2.2) [SUSv4]	initstate(GLIBC_2.2) [SUSv4]	initstate_r(GLIBC_2.2) [LSB]	insque(GLIBC_2.2) [SUSv4]
isatty(GLIBC_2.2) [SUSv4]	isblank(GLIBC_2.2) [SUSv4]	jrand48(GLIBC_2.2) [SUSv4]	jrand48_r(GLIBC_2.2) [LSB]
l64a(GLIBC_2.2) [SUSv4]	labs(GLIBC_2.2) [SUSv4]	lcong48(GLIBC_2.2) [SUSv4]	lcong48_r(GLIBC_2.2) [LSB]
ldiv(GLIBC_2.2) [SUSv4]	lfind(GLIBC_2.2) [SUSv4]	llabs(GLIBC_2.2) [SUSv4]	lldiv(GLIBC_2.2) [SUSv4]
longjmp(GLIBC_2.2) [SUSv4]	lrand48(GLIBC_2.2) [SUSv4]	lrand48_r(GLIBC_2.2) [LSB]	lsearch(GLIBC_2.2) [SUSv4]
makecontext(GLIBC_2.2) [SUSv3]	malloc(GLIBC_2.2) [SUSv4]	memmem(GLIBC_2.2) [LSB]	mkdtemp(GLIBC_2.2) [SUSv4]
mkstemp(GLIBC_2.2) [SUSv4]	mktemp(GLIBC_2.2) [SUSv3]	mrnd48(GLIBC_2.2) [SUSv4]	mrnd48_r(GLIBC_2.2) [LSB]
nftw(GLIBC_2.3) [SUSv4]	nrnd48(GLIBC_2.2) [SUSv4]	nrnd48_r(GLIBC_2.2) [LSB]	ntohl(GLIBC_2.2) [SUSv4]
ntohs(GLIBC_2.2) [SUSv4]	open_memstream(GLIBC_2.2) [SUSv4]	openlog(GLIBC_2.2) [SUSv4]	perror(GLIBC_2.2) [SUSv4]
posix_openpt(GLIBC_2.2.1) [SUSv4]	ptsname(GLIBC_2.2) [SUSv4]	putenv(GLIBC_2.2) [SUSv4]	qsort(GLIBC_2.2) [SUSv4]

rand(GLIBC_2.2) [SUSv4]	rand_r(GLIBC_2.2) [SUSv4]	random(GLIBC_2.2) [SUSv4]	random_r(GLIBC_2.2) [LSB]
realloc(GLIBC_2.2) [SUSv4]	realpath(GLIBC_2.3) [SUSv4]	remque(GLIBC_2.2) [SUSv4]	scandir(GLIBC_2.2) [SUSv4]
scandir64(GLIBC_2.2) [LSB]	seed48(GLIBC_2.2) [SUSv4]	seed48_r(GLIBC_2.2) [LSB]	sendfile(GLIBC_2.2) [LSB]
setenv(GLIBC_2.2) [SUSv4]	sethostname(GLIBC_2.2) [LSB]	setlogmask(GLIBC_2.2) [SUSv4]	setstate(GLIBC_2.2) [SUSv4]
setstate_r(GLIBC_2.2) [LSB]	srand(GLIBC_2.2) [SUSv4]	srand48(GLIBC_2.2) [SUSv4]	srand48_r(GLIBC_2.2) [LSB]
srandom(GLIBC_2.2) [SUSv4]	srandom_r(GLIBC_2.2) [LSB]	strtod(GLIBC_2.2) [SUSv4]	strtol(GLIBC_2.2) [SUSv4]
strtoul(GLIBC_2.2) [SUSv4]	swapcontext(GLIBC_2.2) [SUSv3]	syslog(GLIBC_2.2) [SUSv4]	system(GLIBC_2.2) [LSB]
tdelete(GLIBC_2.2) [SUSv4]	tfind(GLIBC_2.2) [SUSv4]	tmpfile(GLIBC_2.2) [SUSv4]	tmpnam(GLIBC_2.2) [SUSv4]
tsearch(GLIBC_2.2) [SUSv4]	ttynam(GLIBC_2.2) [SUSv4]	ttynam_r(GLIBC_2.2) [SUSv4]	twalk(GLIBC_2.2) [SUSv4]
unlockpt(GLIBC_2.2) [SUSv4]	unsetenv(GLIBC_2.2) [SUSv4]	usleep(GLIBC_2.2) [SUSv3]	verrx(GLIBC_2.2) [LSB]
vfscanf(GLIBC_2.2) [LSB]	vscanf(GLIBC_2.2) [LSB]	vsscanf(GLIBC_2.2) [LSB]	vsyslog(GLIBC_2.2) [LSB]
warn(GLIBC_2.2) [LSB]	warnx(GLIBC_2.2) [LSB]	wordexp(GLIBC_2.2.2) [SUSv4]	wordfree(GLIBC_2.2) [SUSv4]

An LSB conforming implementation shall provide the architecture specific deprecated functions for Standard Library specified in Table 10-33, 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 10-33 libc - Standard Library Deprecated Function Interfaces

basename(GLIBC_2.2) [LSB]	getdomainname(GLIBC_2.2) [LSB]	inet_aton(GLIBC_2.2) [LSB]	tmpnam(GLIBC_2.2) [SUSv4]
---------------------------	--------------------------------	----------------------------	---------------------------

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard Library specified in Table 10-34, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-34 libc - Standard Library Data Interfaces

__environ(GLIBC_2.2) [LSB]	_environ(GLIBC_2.2) [LSB]	_sys_errlist(GLIBC_2.12) [LSB]	environ(GLIBC_2.2) [SUSv4]
getdate_err(GLIBC_2.2) [SUSv4]	optarg(GLIBC_2.2) [SUSv4]	opterr(GLIBC_2.2) [SUSv4]	optind(GLIBC_2.2) [SUSv4]

optopt(GLIBC_2.2) [SUSv4]			
---------------------------	--	--	--

10.2.22 GNU Extensions for libc

10.2.22.1 Interfaces for GNU Extensions for libc

An LSB conforming implementation shall provide the architecture specific functions for GNU Extensions for libc specified in Table 10-35, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-35 libc - GNU Extensions for libc Function Interfaces

gnu_get_libc_release(GLIBC_2.2) [LSB]	gnu_get_libc_version(GLIBC_2.2) [LSB]		
---------------------------------------	---------------------------------------	--	--

10.3 Data Definitions for libc

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

10.3.1 argz.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.2 assert.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.3 cpio.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.4 ctype.h

```
enum {
    _ISupper = 256,
    _ISlower = 512,
    _ISalpha = 1024,
    _ISdigit = 2048,
    _ISxdigit = 4096,
    _ISspace = 8192,
    _ISprint = 16384,
    _ISgraph = 32768,
    _ISblank = 1,
    _IScntrl = 2,
    _ISPunct = 4,
    _ISalnum = 8
};
```

10.3.5 dirent.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.6 elf.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.7 endian.h

```
#define __BYTE_ORDER __LITTLE_ENDIAN
```

10.3.8 errno.h

```
#define EDEADLOCK      EDEADLK
```

10.3.9 fcntl.h

```
#define O_LARGEFILE      0
#define O_DIRECTORY      0200000
#define O_NOFOLLOW       0400000
#define POSIX_FADV_DONTNEED 4
#define POSIX_FADV_NOREUSE 5

#define F_GETLK64        5
#define F_SETLK64        6
#define F_SETLKW64       7
```

10.3.10 fmtmsg.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

```
*/
```

10.3.11 fnmatch.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.12 ftw.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.13 getopt.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.14 glob.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.15 iconv.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.16 ifaddrs.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.17 inttypes.h

```
#define __PRI64_PREFIX "l"
#define __PRIPTR_PREFIX "l"

typedef ldiv_t imaxdiv_t;
```

10.3.18 langinfo.h

```
/*
 * This header is architecture neutral
```

```

* Please refer to the generic specification for details
*/

```

10.3.19 limits.h

```

#define LONG_MAX      0x7FFFFFFFFFFFFFFFL
#define ULONG_MAX     0xFFFFFFFFFFFFFFFFUL
#define LONG_BIT      64

#define CHAR_MAX      SCHAR_MAX
#define CHAR_MIN      SCHAR_MIN

#define PTHREAD_STACK_MIN 196608

```

10.3.20 link.h

```

struct dl_phdr_info {
    Elf64_Addr dlpi_addr;
    const char *dlpi_name;
    const Elf64_Phdr *dlpi_phdr;
    Elf64_Half dlpi_phnum;
    unsigned long long int dlpi_adds;
    unsigned long long int dlpi_subs;
    size_t dlpi_tls_modid;
    void *dlpi_tls_data;
};

```

10.3.21 locale.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.22 lsb/time.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.23 lsb/types.h

```

typedef int64_t ssize_t;

```

10.3.24 lsb/wchar.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.25 net/if.h

```

/*
 * This header is architecture neutral

```

```
* Please refer to the generic specification for details
*/
```

10.3.26 netdb.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.27 netinet/icmp6.h

```
#define ND_NA_FLAG_OVERRIDE      0x00000020
#define ND_NA_FLAG_SOLICITED    0x00000040
#define ND_NA_FLAG_ROUTER       0x00000080
#define ICMP6_RR_RESULT_FLAGS_FORBIDDEN 0x0010
#define ICMP6_RR_RESULT_FLAGS_OOB 0x0020
```

10.3.28 netinet/igmp.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.29 netinet/in.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.30 netinet/in_system.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.31 netinet/ip.h

```
struct timestamp {
    u_int8_t len;
    u_int8_t ptr;
    unsigned int flags:4;
    unsigned int overflow:4;
    u_int32_t data[9];
};
struct iphdr {
    unsigned int ihl:4;
    unsigned int version:4;
    u_int8_t tos;
    u_int16_t tot_len;
    u_int16_t id;
    u_int16_t frag_off;
    u_int8_t ttl;
    u_int8_t protocol;
    u_int16_t check;
```

```

    u_int32_t saddr;
    u_int32_t daddr;
};
struct ip {
    unsigned int ip_hl:4;
    unsigned int ip_v:4;
    u_int8_t ip_tos;
    u_short ip_len;
    u_short ip_id;
    u_short ip_off;
    u_int8_t ip_ttl;
    u_int8_t ip_p;
    u_short ip_sum;
    struct in_addr ip_src;
    struct in_addr ip_dst;
};
struct ip_timestamp {
    u_int8_t ipt_code;
    u_int8_t ipt_len;
    u_int8_t ipt_ptr;
    unsigned int ipt_flg:4;
    unsigned int ipt_oflw:4;
    u_int32_t data[9];
};

```

10.3.32 netinet/ip6.h

```

#define IP6_ALERT_MLD    0x0000
#define IP6F_MORE_FRAG  0x0100
#define IP6_ALERT_RSVP   0x0100
#define IP6_ALERT_AN     0x0200
#define IP6F_RESERVED_MASK 0x0600
#define IP6F_OFF_MASK    0xf8ff

```

10.3.33 netinet/ip_icmp.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.34 netinet/tcp.h

```

struct tcphdr {
    uint16_t source;
    uint16_t dest;
    uint32_t seq;
    uint32_t ack_seq;
    uint16_t res1:4;
    uint16_t doff:4;
    uint16_t fin:1;
    uint16_t syn:1;
    uint16_t rst:1;
    uint16_t psh:1;
    uint16_t ack:1;
    uint16_t urg:1;
    uint16_t res2:2;
    uint16_t window;
    uint16_t check;
    uint16_t urg_ptr;
};

```

10.3.35 netinet/udp.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.36 nl_types.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.37 pwd.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.38 regex.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.39 rpc/auth.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.40 rpc/clnt.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.41 rpc/rpc_msg.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.42 rpc/svc.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.43 rpc/types.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.44 rpc/xdr.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.45 sched.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.46 search.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.47 setjmp.h

```

typedef long int __jmp_buf[70] __attribute__((aligned(16)));

```

10.3.48 signal.h

```

#define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int))-4)

#define SI_PAD_SIZE ((SI_MAX_SIZE/sizeof(int))-4)

struct sigaction {
    union {
        sighandler_t _sa_handler;
        void (*_sa_sigaction) (int, siginfo_t *, void *);
    } __sigaction_handler;
    unsigned long int sa_flags;
    sigset_t sa_mask; /* mask last for extensibility */
};

#define MINSIGSTKSZ 131027 /* Minimum stack size for a signal
handler. */
#define SIGSTKSZ 262144 /* System default stack size. */

struct ia64_fpreg {
    union {
        unsigned long int bits[2];
        long double __dummy; /* force 16-byte alignment */
    } u;
};

```

```

struct sigcontext {
    unsigned long int sc_flags;
    unsigned long int sc_nat;
    stack_t sc_stack;
    unsigned long int sc_ip;
    unsigned long int sc_cfm;
    unsigned long int sc_um;
    unsigned long int sc_ar_rsc;
    unsigned long int sc_ar_bsp;
    unsigned long int sc_ar_rnat;
    unsigned long int sc_ar_ccv;
    unsigned long int sc_ar_unat;
    unsigned long int sc_ar_fpsr;
    unsigned long int sc_ar_pfs;
    unsigned long int sc_ar_lc;
    unsigned long int sc_pr;
    unsigned long int sc_br[8];
    unsigned long int sc_gr[32];
    struct ia64_fpreg sc_fr[128];
    unsigned long int sc_rbs_base; /* NULL or new base of
sighandler's rbs */
    unsigned long int sc_loadrs; /* see description above */
    unsigned long int sc_ar25; /* cmp8xchg16 uses this */
    unsigned long int sc_ar26; /* rsvd for scratch use */
    unsigned long int sc_rsvd[12];
    unsigned long int sc_mask; /* really sigset_t, but unsigned
long for convenience at the us */
};

```

10.3.49 spawn.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.50 stddef.h

```

typedef int wchar_t;
typedef unsigned long int size_t;
typedef long int ptrdiff_t;

```

10.3.51 stdint.h

```

#define INT64_C(c)      c ## L
#define INTMAX_C(c)    c ## L
#define __INT64_C(c)   c ## L
#define UINT64_C(c)    c ## UL
#define UINTMAX_C(c)   c ## UL
#define __UINT64_C(c)  c ## UL

#define INTPTR_MIN      (-9223372036854775807L-1)
#define INT_FAST16_MIN  (-9223372036854775807L-1)
#define INT_FAST32_MIN  (-9223372036854775807L-1)
#define PTRDIFF_MIN     (-9223372036854775807L-1)
#define SIZE_MAX        (18446744073709551615UL)
#define UINTPTR_MAX      (18446744073709551615UL)
#define UINT_FAST16_MAX  (18446744073709551615UL)
#define UINT_FAST32_MAX  (18446744073709551615UL)
#define INTPTR_MAX       (9223372036854775807L)
#define INT_FAST16_MAX   (9223372036854775807L)
#define INT_FAST32_MAX   (9223372036854775807L)

```

```

#define PTRDIFF_MAX      (9223372036854775807L)

typedef long int int64_t;
typedef long int intmax_t;
typedef unsigned long int uintmax_t;
typedef long int intptr_t;
typedef unsigned long int uintptr_t;
typedef unsigned long int uint64_t;
typedef long int int_least64_t;
typedef unsigned long int uint_least64_t;
typedef long int int_fast16_t;
typedef long int int_fast32_t;
typedef long int int_fast64_t;
typedef unsigned long int uint_fast16_t;
typedef unsigned long int uint_fast32_t;
typedef unsigned long int uint_fast64_t;

```

10.3.52 stdio.h

```

#define __IO_FILE_SIZE 216

```

10.3.53 stdlib.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.54 string.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.55 sys/epoll.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.56 sys/file.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.57 sys/inotify.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.58 sys/io.h

```
extern int ioperm(unsigned long int from, unsigned long int num,
                  int turn_on);
extern int iopl(int level);
```

10.3.59 sys/ioctl.h

```
#define TIOCGWINSZ      0x5413
#define TIOCSWINSZ      0x5414
#define FIONREAD        0x541B
#define TIOCNOTTY       0x5422
```

10.3.60 sys/ipc.h

```
struct ipc_perm {
    key_t __key;           /* Key. */
    uid_t uid;             /* Owner's user ID. */
    gid_t gid;             /* Owner's group ID. */
    uid_t cuid;            /* Creator's user ID. */
    uid_t cgid;            /* Creator's group ID. */
    mode_t mode;           /* Read/write permission. */
    unsigned short __seq;  /* Sequence number. */
    unsigned short __pad1;
    unsigned long int __unused1;
    unsigned long int __unused2;
};
```

10.3.61 sys/mman.h

```
#define MCL_CURRENT      1
#define MCL_FUTURE       2
```

10.3.62 sys/msg.h

```
struct msqid_ds {
    struct ipc_perm msg_perm; /* structure describing operation
    permission */
    time_t msg_stime;         /* time of last msgsnd command */
    time_t msg_rtime;         /* time of last msgrcv command */
    time_t msg_ctime;         /* time of last change */
    unsigned long int __msg_cbytes; /* current number of bytes
    on queue */
    unsigned long int msg_qnum; /* number of messages currently on
    queue */
    unsigned long int msg_qbytes; /* max number of bytes
    allowed on queue */
    pid_t msg_lspid;          /* pid of last msgsnd() */
    pid_t msg_lrpid;          /* pid of last msgrcv() */
    unsigned long int __unused1;
    unsigned long int __unused2;
};
```

10.3.63 sys/param.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

```
*/
```

10.3.64 sys/poll.h

```
/*
```

```
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.65 sys/ptrace.h

```
enum __ptrace_request {
    PTRACE_TRACEME = 0,
    PTRACE_PEEKTEXT = 1,
    PTRACE_PEEKDATA = 2,
    PTRACE_PEEKUSER = 3,
    PTRACE_POKEUSER = 4,
    PTRACE_POKEUSER = 5,
    PTRACE_POKEUSER = 6,
    PTRACE_CONT = 7,
    PTRACE_KILL = 8,
    PTRACE_SINGLESTEP = 9,
    PTRACE_ATTACH = 16,
    PTRACE_DETACH = 17,
    PTRACE_SYSCALL = 24,
    PTRACE_SETOPTIONS = 0x4200,
    PTRACE_GETEVENTMSG = 0x4201,
    PTRACE_GETSIGINFO = 0x4202,
    PTRACE_SETSIGINFO = 0x4203
};
```

10.3.66 sys/resource.h

```
/*
```

```
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.67 sys/select.h

```
/*
```

```
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.68 sys/sem.h

```
struct semid_ds {
    struct ipc_perm sem_perm; /* operation permission struct */
    time_t sem_otime; /* last semop() time */
    time_t sem_ctime; /* last time changed by semctl() */
    unsigned long int sem_nsems; /* number of semaphores in
set */
    unsigned long int __unused1;
    unsigned long int __unused2;
};
```

10.3.69 sys/shm.h

```

#define SHMLBA (1024*1024)

struct shmid_ds {
    struct ipc_perm shm_perm; /* operation permission struct */
    size_t shm_segsz; /* size of segment in bytes */
    time_t shm_atime; /* time of last shmat() */
    time_t shm_dtime; /* time of last shmdt() */
    time_t shm_ctime; /* time of last change by shmctl() */
    /*
    pid_t shm_cpid; /* pid of creator */
    pid_t shm_lpid; /* pid of last shmop */
    unsigned long int shm_nattch; /* number of current attaches
    */
    unsigned long int __unused1;
    unsigned long int __unused2;
};

```

10.3.70 sys/socket.h

```

typedef uint64_t __ss_aligntype;

#define SO_RCVLOWAT 18
#define SO_SNDLOWAT 19
#define SO_RCVTIMEO 20
#define SO_SNDTIMEO 21

```

10.3.71 sys/stat.h

```

#define _MKNOD_VER 0
#define _STAT_VER 1

struct stat {
    dev_t st_dev;
    ino_t st_ino;
    nlink_t st_nlink;
    mode_t st_mode;
    uid_t st_uid;
    gid_t st_gid;
    unsigned int pad0;
    dev_t st_rdev;
    off_t st_size;
    struct timespec st_atim; /* Time of last access. */
    struct timespec st_mtim; /* Time of last modification. */
    struct timespec st_ctim; /* Time of last status change. */
    blksize_t st_blksize;
    blkcnt_t st_blocks;
    unsigned long int __unused[3];
};

struct stat64 {
    dev_t st_dev;
    ino64_t st_ino;
    nlink_t st_nlink;
    mode_t st_mode;
    uid_t st_uid;
    gid_t st_gid;
    unsigned int pad0;
    dev_t st_rdev;
    off_t st_size;
    struct timespec st_atim; /* Time of last access. */
    struct timespec st_mtim; /* Time of last modification. */

```

```

    struct timespec st_ctim;    /* Time of last status change. */
    blksize_t st_blksize;
    blkcnt64_t st_blocks;
    unsigned long int __unused[3];
};

```

10.3.72 sys/statfs.h

```

struct statfs {
    long int f_type;           /* type of filesystem */
    long int f_bsize;          /* optimal transfer block size */
    fsblkcnt_t f_blocks;       /* total data blocks in file system */
    /*
    fsblkcnt_t f_bfree;         /* free blocks in fs */
    fsblkcnt_t f_bavail;       /* free blocks avail to non-superuser */
    */
    fsfilcnt_t f_files;        /* total file nodes in file system */
    /*
    fsfilcnt_t f_ffree;         /* free file nodes in file system */
    fsid_t f_fsid;             /* file system id */
    long int f_namelen;        /* maximum length of filenames */
    long int f_frsize;         /* fragment size */
    long int f_spare[5];       /* spare for later */
};
struct statfs64 {
    long int f_type;           /* type of filesystem */
    long int f_bsize;          /* optimal transfer block size */
    fsblkcnt64_t f_blocks;     /* total data blocks in file system */
    /*
    fsblkcnt64_t f_bfree;       /* free blocks in fs */
    fsblkcnt64_t f_bavail;      /* free blocks avail to non-
superuser */
    fsfilcnt64_t f_files;      /* total file nodes in file system */
    /*
    fsfilcnt64_t f_ffree;       /* free file nodes in file system */
    fsid_t f_fsid;             /* file system id */
    long int f_namelen;        /* maximum length of filenames */
    long int f_frsize;         /* fragment size */
    long int f_spare[5];       /* spare for later */
};

```

10.3.73 sys/statvfs.h

```

struct statvfs {
    unsigned long int f_bsize;
    unsigned long int f_frsize;
    fsblkcnt64_t f_blocks;
    fsblkcnt64_t f_bfree;
    fsblkcnt64_t f_bavail;
    fsfilcnt64_t f_files;
    fsfilcnt64_t f_ffree;
    fsfilcnt64_t f_favail;
    unsigned long int f_fsid;
    unsigned long int f_flag;
    unsigned long int f_namemax;
    unsigned int __f_spare[6];
};
struct statvfs64 {
    unsigned long int f_bsize;
    unsigned long int f_frsize;
    fsblkcnt64_t f_blocks;
    fsblkcnt64_t f_bfree;
    fsblkcnt64_t f_bavail;

```

```

    fsfilcnt64_t f_files;
    fsfilcnt64_t f_ffree;
    fsfilcnt64_t f_favail;
    unsigned long int f_fsid;
    unsigned long int f_flag;
    unsigned long int f_namemax;
    unsigned int __f_spare[6];
};

```

10.3.74 sys/sysinfo.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.75 sys/time.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.76 sys/timeb.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.77 sys/times.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.78 sys/un.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.79 sys/utsname.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.80 sys/wait.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details

```

*/

10.3.81 sysexits.h

/*

* This header is architecture neutral
 * Please refer to the generic specification for details
 */

10.3.82 syslog.h

/*

* This header is architecture neutral
 * Please refer to the generic specification for details
 */

10.3.83 tar.h

/*

* This header is architecture neutral
 * Please refer to the generic specification for details
 */

10.3.84 termios.h

```
#define OLCUC      0000002
#define ONLCR      0000004
#define XCASE      0000004
#define NLDLY      0000400
#define CR1        0001000
#define IUCLC      0001000
#define CR2        0002000
#define CR3        0003000
#define CRDLY      0003000
#define TAB1       0004000
#define TAB2       0010000
#define TAB3       0014000
#define TABDLY     0014000
#define BS1        0020000
#define BSDLY      0020000
#define VT1        0040000
#define VTDLY      0040000
#define FF1        0100000
#define FFDLY      0100000

#define VSUSP      10
#define VEOL       11
#define VREPRINT   12
#define VDISCARD   13
#define VWERASE    14
#define VEOL2      16
#define VMIN       6
#define VSWTC      7
#define VSTART     8
#define VSTOP      9

#define IXON       0002000
#define IXOFF      0010000

#define CS6        0000020
```

```

#define CS7      0000040
#define CS8      0000060
#define CSIZE    0000060
#define CSTOPB   0000100
#define CREAD    0000200
#define PARENB   0000400
#define PARODD   0001000
#define HUPCL    0002000
#define CLOCAL   0004000
#define VTIME    5

#define ISIG      0000001
#define ICANON    0000002
#define ECHOE     0000020
#define ECHOK     0000040
#define ECHONL    0000100
#define NOFLSH    0000200
#define TOSTOP    0000400
#define ECHOCTL   0001000
#define ECHOPRT   0002000
#define ECHOKE    0004000
#define FLUSHO    0010000
#define PENDIN    0040000
#define IEXTEN    0100000

```

10.3.85 time.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.3.86 ucontext.h

```

#define rPOS      r16
#define rTMP      r16
#define rCPOS     r17
#define rB5       r18
#define rNAT      r18
#define rB4       r19
#define rB3       r20
#define rB2       r21
#define rB1       r22
#define rB0       r23
#define rRSC      r24
#define rBSP      r25
#define rRNAT     r26
#define rUNAT     r27
#define rFPSR     r28
#define rPFS      r29
#define rLC       r30
#define rPR       r31
#define _SC_GR0_OFFSET \
    (((char *) &((struct sigcontext *) 0)->sc_gr[0]) - (char *)
0)

typedef struct sigcontext mcontext_t;

#define uc_mcontext      _u._mc
#define uc_sigmask       _u._mc.sc_mask
#define uc_stack         _u._mc.sc_stack
#define uc_link _u._uc._link

```

```
typedef struct ucontext {
    union {
        mcontext_t _mc;
        struct {
            unsigned long int _pad[_SC_GRO_OFFSET / 8];
            struct ucontext *_link;
        } _uc;
    } _u;
} ucontext_t;
```

10.3.87 ulimit.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.88 unistd.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.89 utime.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.3.90 utmp.h

```
struct lastlog {
    time_t ll_time;
    char ll_line[UT_LINESIZE];
    char ll_host[UT_HOSTSIZE];
};

struct utmp {
    short ut_type;           /* Type of login. */
    pid_t ut_pid;           /* Process ID of login process. */
    char ut_line[UT_LINESIZE]; /* Devicename. */
    char ut_id[4];          /* Inittab ID. */
    char ut_user[UT_NAMESIZE]; /* Username. */
    char ut_host[UT_HOSTSIZE]; /* Hostname for remote login. */
    struct exit_status ut_exit; /* Exit status of a process marked
as DEAD_PROCESS. */
    long int ut_session;     /* Session ID, used for windowing. */
    struct timeval ut_tv;    /* Time entry was made. */
    int32_t ut_addr_v6[4];  /* Internet address of remote host. */
    char __unused[20];      /* Reserved for future use. */
};
```

10.3.91 utmpx.h

```
struct utmpx {
```

IECNORM.COM: Click to view the full PDF of ISO/IEC 23360-3-2:2021

```

    short ut_type;           /* Type of login. */
    pid_t ut_pid;           /* Process ID of login process. */
    char ut_line[UT_LINESIZE]; /* Devicename. */
    char ut_id[4];          /* Inittab ID. */
    char ut_user[UT_NAMESIZE]; /* Username. */
    char ut_host[UT_HOSTSIZE]; /* Hostname for remote login. */
    struct exit_status ut_exit; /* Exit status of a process marked
as DEAD_PROCESS. */
    long int ut_session;     /* Session ID, used for windowing.
*/
    struct timeval ut_tv;    /* Time entry was made. */
    int32_t ut_addr_v6[4];  /* Internet address of remote host.
*/
    char __unused[20];      /* Reserved for future use. */
};

```

10.3.92 wordexp.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

10.4 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 10.2 shall behave as described in the referenced base document. For interfaces referencing LSB and not listed below, please see the generic part of the specification.

ioperm

Name

ioperm — set port input/output permissions

Synopsis

```
#include <sys/io.h> /* for glibc */
```

```
int ioperm(unsigned long from, unsigned long num, int turn_on);
```

Description

ioperm sets the port access permission bits for the process for num bytes starting from port address from to the value turn_on. The use of ioperm requires root privileges.

Only the first 0x3ff I/O ports can be specified in this manner. For more ports, the iopl function must be used. Permissions are not inherited on fork, but on exec they are. This is useful for giving port access permissions to non-privileged tasks.

Return Value

On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

Notes

Libc5 treats it as a system call and has a prototype in <unistd.h>. Glibc1 does not have a prototype. Glibc2 has a prototype both in <sys/io.h> and in <sys/perm.h>. Avoid the latter, it is available on i386 only.

iopl

Name

iopl — change I/O privilege level

Synopsis

```
#include <sys/io.h> /* for glibc */
```

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

```
int iopl(int level);
```

Description

`ioctl` changes the I/O privilege level of the current process, as specified in `level`.

This call is necessary to allow 8514-compatible X servers to run under Linux. Since these X servers require access to all 65536 I/O ports, the `ioperm` call is not sufficient.

In addition to granting unrestricted I/O port access, running at a higher I/O privilege level also allows the process to disable interrupts. This will probably crash the system, and is not recommended.

Permissions are inherited by `fork` and `exec`.

The I/O privilege level for a normal process is 0.

Return Value

On success, zero is returned. On error, -1 is returned, and `errno` is set appropriately.

Errors

EINVAL

`level` is greater than 3.

EPERM

The current user is not the super-user.

10.5 Interfaces for libm

Table 10-36 defines the library name and shared object name for the `libm` library

Table 10-36 libm Definition

Library:	libm
SONAME:	libm.so.6.1

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

[LSB] LSB Core - Generic

[SUSv3] POSIX 1003.1-2001 (ISO/IEC 9945-2003)

[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

10.5.1 Math

10.5.1.1 Interfaces for Math

An LSB conforming implementation shall provide the architecture specific functions for Math specified in Table 10-37, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-37 libm - Math Function Interfaces

<code>__finite(GLIBC_2.2)</code> [LSB]	<code>__finitef(GLIBC_2.2)</code> [LSB]	<code>__finitel(GLIBC_2.2)</code> [LSB]	<code>__fpclassify(GLIBC_2.2)</code> [LSB]
--	---	---	--

__fpclassifyf(GLIBC_2.2) [LSB]	__fpclassifyf(GLIBC_2.2) [LSB]	__signbit(GLIBC_2.2) [LSB]	__signbitf(GLIBC_2.2) [LSB]
__signbitl(GLIBC_2.2) [LSB]	acos(GLIBC_2.2) [SUSv4]	acosf(GLIBC_2.2) [SUSv4]	acosh(GLIBC_2.2) [SUSv4]
acoshf(GLIBC_2.2) [SUSv4]	acoshl(GLIBC_2.2) [SUSv4]	acosl(GLIBC_2.2) [SUSv4]	asin(GLIBC_2.2) [SUSv4]
asinf(GLIBC_2.2) [SUSv4]	asinh(GLIBC_2.2) [SUSv4]	asinhf(GLIBC_2.2) [SUSv4]	asinhf(GLIBC_2.2) [SUSv4]
asinl(GLIBC_2.2) [SUSv4]	atan(GLIBC_2.2) [SUSv4]	atan2(GLIBC_2.2) [SUSv4]	atan2f(GLIBC_2.2) [SUSv4]
atan2l(GLIBC_2.2) [SUSv4]	atanf(GLIBC_2.2) [SUSv4]	atanh(GLIBC_2.2) [SUSv4]	atanhf(GLIBC_2.2) [SUSv4]
atanhl(GLIBC_2.2) [SUSv4]	atanl(GLIBC_2.2) [SUSv4]	cabs(GLIBC_2.2) [SUSv4]	cabsf(GLIBC_2.2) [SUSv4]
cabsf(GLIBC_2.2) [SUSv4]	cacos(GLIBC_2.2) [SUSv4]	cacosf(GLIBC_2.2) [SUSv4]	cacosh(GLIBC_2.2) [SUSv4]
cacoshf(GLIBC_2.2) [SUSv4]	cacoshl(GLIBC_2.2) [SUSv4]	cacosl(GLIBC_2.2) [SUSv4]	carg(GLIBC_2.2) [SUSv4]
cargf(GLIBC_2.2) [SUSv4]	cargl(GLIBC_2.2) [SUSv4]	casin(GLIBC_2.2) [SUSv4]	casinf(GLIBC_2.2) [SUSv4]
casinh(GLIBC_2.2) [SUSv4]	casinhf(GLIBC_2.2) [SUSv4]	casinhf(GLIBC_2.2) [SUSv4]	casinl(GLIBC_2.2) [SUSv4]
catan(GLIBC_2.2) [SUSv4]	catanf(GLIBC_2.2) [SUSv4]	catanh(GLIBC_2.2) [SUSv4]	catanhf(GLIBC_2.2) [SUSv4]
catanhf(GLIBC_2.2) [SUSv4]	catanl(GLIBC_2.2) [SUSv4]	cbrt(GLIBC_2.2) [SUSv4]	cbrtf(GLIBC_2.2) [SUSv4]
cbrtl(GLIBC_2.2) [SUSv4]	ccos(GLIBC_2.2) [SUSv4]	ccosf(GLIBC_2.2) [SUSv4]	ccosh(GLIBC_2.2) [SUSv4]
ccoshf(GLIBC_2.2) [SUSv4]	ccoshl(GLIBC_2.2) [SUSv4]	ccosl(GLIBC_2.2) [SUSv4]	ceil(GLIBC_2.2) [SUSv4]
ceilf(GLIBC_2.2) [SUSv4]	ceilf(GLIBC_2.2) [SUSv4]	cexp(GLIBC_2.2) [SUSv4]	cexpf(GLIBC_2.2) [SUSv4]
cexpl(GLIBC_2.2) [SUSv4]	cimag(GLIBC_2.2) [SUSv4]	cimagf(GLIBC_2.2) [SUSv4]	cimagl(GLIBC_2.2) [SUSv4]
clog(GLIBC_2.2) [SUSv4]	clog10(GLIBC_2.2) [LSB]	clog10f(GLIBC_2.2) [LSB]	clog10l(GLIBC_2.2) [LSB]
clogf(GLIBC_2.2) [SUSv4]	clogl(GLIBC_2.2) [SUSv4]	conj(GLIBC_2.2) [SUSv4]	conjf(GLIBC_2.2) [SUSv4]
conjl(GLIBC_2.2) [SUSv4]	copysign(GLIBC_2.2) [SUSv4]	copysignf(GLIBC_2.2) [SUSv4]	copysignl(GLIBC_2.2) [SUSv4]
cos(GLIBC_2.2) [SUSv4]	cosf(GLIBC_2.2) [SUSv4]	cosh(GLIBC_2.2) [SUSv4]	coshf(GLIBC_2.2) [SUSv4]

coshl(GLIBC_2.2) [SUSv4]	cosl(GLIBC_2.2) [SUSv4]	cpow(GLIBC_2.2) [SUSv4]	cpowf(GLIBC_2.2) [SUSv4]
cpowl(GLIBC_2.2) [SUSv4]	cproj(GLIBC_2.2) [SUSv4]	cprojf(GLIBC_2.2) [SUSv4]	cprojl(GLIBC_2.2) [SUSv4]
creal(GLIBC_2.2) [SUSv4]	crealf(GLIBC_2.2) [SUSv4]	creall(GLIBC_2.2) [SUSv4]	csin(GLIBC_2.2) [SUSv4]
csinf(GLIBC_2.2) [SUSv4]	csinh(GLIBC_2.2) [SUSv4]	csinhf(GLIBC_2.2) [SUSv4]	csinhl(GLIBC_2.2) [SUSv4]
csinl(GLIBC_2.2) [SUSv4]	csqrt(GLIBC_2.2) [SUSv4]	csqrtf(GLIBC_2.2) [SUSv4]	csqrtl(GLIBC_2.2) [SUSv4]
ctan(GLIBC_2.2) [SUSv4]	ctanf(GLIBC_2.2) [SUSv4]	ctanh(GLIBC_2.2) [SUSv4]	ctanhf(GLIBC_2.2) [SUSv4]
ctanhl(GLIBC_2.2) [SUSv4]	ctanl(GLIBC_2.2) [SUSv4]	drem(GLIBC_2.2) [LSB]	dremf(GLIBC_2.2) [LSB]
dreml(GLIBC_2.2) [LSB]	erf(GLIBC_2.2) [SUSv4]	erfc(GLIBC_2.2) [SUSv4]	erfcf(GLIBC_2.2) [SUSv4]
erfcl(GLIBC_2.2) [SUSv4]	erff(GLIBC_2.2) [SUSv4]	erfl(GLIBC_2.2) [SUSv4]	exp(GLIBC_2.2) [SUSv4]
exp10(GLIBC_2.2) [LSB]	exp10f(GLIBC_2.2) [LSB]	exp10l(GLIBC_2.2) [LSB]	exp2(GLIBC_2.2) [SUSv4]
exp2f(GLIBC_2.2) [SUSv4]	exp2l(GLIBC_2.2) [SUSv4]	expf(GLIBC_2.2) [SUSv4]	expl(GLIBC_2.2) [SUSv4]
expm1(GLIBC_2.2) [SUSv4]	expm1f(GLIBC_2.2) [SUSv4]	expm1l(GLIBC_2.2) [SUSv4]	fabs(GLIBC_2.2) [SUSv4]
fabsf(GLIBC_2.2) [SUSv4]	fabsl(GLIBC_2.2) [SUSv4]	fdim(GLIBC_2.2) [SUSv4]	fdimf(GLIBC_2.2) [SUSv4]
fdiml(GLIBC_2.2) [SUSv4]	feclearexcept(GLIBC_2.2) [SUSv4]	fedisableexcept(GLIBC_2.2) [LSB]	feenableexcept(GLIBC_2.2) [LSB]
fegetenv(GLIBC_2.2) [SUSv4]	fegetexcept(GLIBC_2.2) [LSB]	fegetexceptflag(GLIBC_2.2) [SUSv4]	fegetround(GLIBC_2.2) [SUSv4]
feholdexcept(GLIBC_2.2) [SUSv4]	feraiseexcept(GLIBC_2.2) [SUSv4]	fesetenv(GLIBC_2.2) [SUSv4]	fesetexceptflag(GLIBC_2.2) [SUSv4]
fesetround(GLIBC_2.2) [SUSv4]	fetestexcept(GLIBC_2.2) [SUSv4]	feupdateenv(GLIBC_2.2) [SUSv4]	finite(GLIBC_2.2) [LSB]
finitef(GLIBC_2.2) [LSB]	finitel(GLIBC_2.2) [LSB]	floor(GLIBC_2.2) [SUSv4]	floorf(GLIBC_2.2) [SUSv4]
floorl(GLIBC_2.2) [SUSv4]	fma(GLIBC_2.2) [SUSv4]	fmaf(GLIBC_2.2) [SUSv4]	fmal(GLIBC_2.2) [SUSv4]
fmax(GLIBC_2.2) [SUSv4]	fmaxf(GLIBC_2.2) [SUSv4]	fmaxl(GLIBC_2.2) [SUSv4]	fmin(GLIBC_2.2) [SUSv4]

fminf(GLIBC_2.2) [SUSv4]	fminl(GLIBC_2.2) [SUSv4]	fmod(GLIBC_2.2) [SUSv4]	fmodf(GLIBC_2.2) [SUSv4]
fmodl(GLIBC_2.2) [SUSv4]	frexp(GLIBC_2.2) [SUSv4]	frexpf(GLIBC_2.2) [SUSv4]	frexpl(GLIBC_2.2) [SUSv4]
gamma(GLIBC_2.2) [LSB]	gammaf(GLIBC_2.2) [LSB]	gammal(GLIBC_2.2) [LSB]	hypot(GLIBC_2.2) [SUSv4]
hypotf(GLIBC_2.2) [SUSv4]	hypotl(GLIBC_2.2) [SUSv4]	ilogb(GLIBC_2.2) [SUSv4]	ilogbf(GLIBC_2.2) [SUSv4]
ilogbl(GLIBC_2.2) [SUSv4]	j0(GLIBC_2.2) [SUSv4]	j0f(GLIBC_2.2) [LSB]	j0l(GLIBC_2.2) [LSB]
j1(GLIBC_2.2) [SUSv4]	j1f(GLIBC_2.2) [LSB]	j1l(GLIBC_2.2) [LSB]	jn(GLIBC_2.2) [SUSv4]
jnf(GLIBC_2.2) [LSB]	jnl(GLIBC_2.2) [LSB]	ldexp(GLIBC_2.2) [SUSv4]	ldexpf(GLIBC_2.2) [SUSv4]
ldexpl(GLIBC_2.2) [SUSv4]	lgamma(GLIBC_2.2) [SUSv4]	lgamma_r(GLIBC_2.2) [LSB]	lgammaf(GLIBC_2.2) [SUSv4]
lgammaf_r(GLIBC_2.2) [LSB]	lgammal(GLIBC_2.2) [SUSv4]	lgammal_r(GLIBC_2.2) [LSB]	llrint(GLIBC_2.2) [SUSv4]
llrintf(GLIBC_2.2) [SUSv4]	llrintl(GLIBC_2.2) [SUSv4]	llround(GLIBC_2.2) [SUSv4]	llroundf(GLIBC_2.2) [SUSv4]
llroundl(GLIBC_2.2) [SUSv4]	log(GLIBC_2.2) [SUSv4]	log10(GLIBC_2.2) [SUSv4]	log10f(GLIBC_2.2) [SUSv4]
log10l(GLIBC_2.2) [SUSv4]	log1p(GLIBC_2.2) [SUSv4]	log1pf(GLIBC_2.2) [SUSv4]	log1pl(GLIBC_2.2) [SUSv4]
log2(GLIBC_2.2) [SUSv4]	log2f(GLIBC_2.2) [SUSv4]	log2l(GLIBC_2.2) [SUSv4]	logb(GLIBC_2.2) [SUSv4]
logbf(GLIBC_2.2) [SUSv4]	logbl(GLIBC_2.2) [SUSv4]	logf(GLIBC_2.2) [SUSv4]	logl(GLIBC_2.2) [SUSv4]
lrint(GLIBC_2.2) [SUSv4]	lrintf(GLIBC_2.2) [SUSv4]	lrintl(GLIBC_2.2) [SUSv4]	lround(GLIBC_2.2) [SUSv4]
lroundf(GLIBC_2.2) [SUSv4]	lroundl(GLIBC_2.2) [SUSv4]	matherr(GLIBC_2.2) [LSB]	modf(GLIBC_2.2) [SUSv4]
modff(GLIBC_2.2) [SUSv4]	modfl(GLIBC_2.2) [SUSv4]	nan(GLIBC_2.2) [SUSv4]	nanf(GLIBC_2.2) [SUSv4]
nanl(GLIBC_2.2) [SUSv4]	nearbyint(GLIBC_2.2) [SUSv4]	nearbyintf(GLIBC_2.2) [SUSv4]	nearbyintl(GLIBC_2.2) [SUSv4]
nextafter(GLIBC_2.2) [SUSv4]	nextafterf(GLIBC_2.2) [SUSv4]	nextafterl(GLIBC_2.2) [SUSv4]	nexttoward(GLIBC_2.2) [SUSv4]
nexttowardf(GLIBC_2.2) [SUSv4]	nexttowardl(GLIBC_2.2) [SUSv4]	pow(GLIBC_2.2) [SUSv4]	pow10(GLIBC_2.2) [LSB]
pow10f(GLIBC_2.2) [LSB]	pow10l(GLIBC_2.2) [LSB]	powf(GLIBC_2.2) [SUSv4]	powl(GLIBC_2.2) [SUSv4]

remainder(GLIBC_2.2) [SUSv4]	remainderf(GLIBC_2.2) [SUSv4]	remainderl(GLIBC_2.2) [SUSv4]	remquo(GLIBC_2.2) [SUSv4]
remquof(GLIBC_2.2) [SUSv4]	remquol(GLIBC_2.2) [SUSv4]	rint(GLIBC_2.2) [SUSv4]	rintf(GLIBC_2.2) [SUSv4]
rintl(GLIBC_2.2) [SUSv4]	round(GLIBC_2.2) [SUSv4]	roundf(GLIBC_2.2) [SUSv4]	roundl(GLIBC_2.2) [SUSv4]
scalb(GLIBC_2.2) [SUSv3]	scalbf(GLIBC_2.2) [LSB]	scalbl(GLIBC_2.2) [LSB]	scalbln(GLIBC_2.2) [SUSv4]
scalblnf(GLIBC_2.2) [SUSv4]	scalblnl(GLIBC_2.2) [SUSv4]	scalbn(GLIBC_2.2) [SUSv4]	scalbnf(GLIBC_2.2) [SUSv4]
scalbnl(GLIBC_2.2) [SUSv4]	significand(GLIBC_2.2) [LSB]	significandf(GLIBC_2.2) [LSB]	significandl(GLIBC_2.2) [LSB]
sin(GLIBC_2.2) [SUSv4]	sincos(GLIBC_2.2) [LSB]	sincosf(GLIBC_2.2) [LSB]	sincosl(GLIBC_2.2) [LSB]
sinf(GLIBC_2.2) [SUSv4]	sinh(GLIBC_2.2) [SUSv4]	sinhf(GLIBC_2.2) [SUSv4]	sinhl(GLIBC_2.2) [SUSv4]
sinl(GLIBC_2.2) [SUSv4]	sqrt(GLIBC_2.2) [SUSv4]	sqrtrf(GLIBC_2.2) [SUSv4]	sqrtrl(GLIBC_2.2) [SUSv4]
tan(GLIBC_2.2) [SUSv4]	tanf(GLIBC_2.2) [SUSv4]	tanh(GLIBC_2.2) [SUSv4]	tanhf(GLIBC_2.2) [SUSv4]
tanhf(GLIBC_2.2) [SUSv4]	tanl(GLIBC_2.2) [SUSv4]	tgamma(GLIBC_2.2) [SUSv4]	tgammaf(GLIBC_2.2) [SUSv4]
tgammaf(GLIBC_2.2) [SUSv4]	trunc(GLIBC_2.2) [SUSv4]	truncf(GLIBC_2.2) [SUSv4]	truncl(GLIBC_2.2) [SUSv4]
y0(GLIBC_2.2) [SUSv4]	y0f(GLIBC_2.2) [LSB]	y0l(GLIBC_2.2) [LSB]	y1(GLIBC_2.2) [SUSv4]
y1f(GLIBC_2.2) [LSB]	y1l(GLIBC_2.2) [LSB]	yn(GLIBC_2.2) [SUSv4]	ynf(GLIBC_2.2) [LSB]
ynl(GLIBC_2.2) [LSB]			

An LSB conforming implementation shall provide the architecture specific deprecated functions for Math specified in Table 10-38, 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 10-38 libm - Math Deprecated Function Interfaces

drem(GLIBC_2.2) [LSB]	dremf(GLIBC_2.2) [LSB]	dreml(GLIBC_2.2) [LSB]	finite(GLIBC_2.2) [LSB]
finitef(GLIBC_2.2) [LSB]	finitel(GLIBC_2.2) [LSB]	gamma(GLIBC_2.2) [LSB]	gammaf(GLIBC_2.2) [LSB]

gammal(GLIBC_2.2) [LSB]	matherr(GLIBC_2.2) [LSB]		
-------------------------	--------------------------	--	--

An LSB conforming implementation shall provide the architecture specific data interfaces for Math specified in Table 10-39, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-39 libm - Math Data Interfaces

signgam(GLIBC_2.2) [SUSv4]			
----------------------------	--	--	--

10.6 Data Definitions for libm

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

10.6.1 complex.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.6.2 fenv.h

```
#define FE_INVALID      (1UL << 0)
#define FE_DIVBYZERO   (1UL << 2)
#define FE_OVERFLOW    (1UL << 3)
#define FE_UNDERFLOW   (1UL << 4)
#define FE_INEXACT      (1UL << 5)
#define FE_UNNORMAL    1UL << 1

#define FE_ALL_EXCEPT \
    (FE_INEXACT | FE_UNDERFLOW | FE_OVERFLOW | FE_DIVBYZERO | \
FE_UNNORMAL | FE_INVALID)

#define FE_TONEAREST    0
#define FE_DOWNWARD     1
#define FE_UPWARD       2
#define FE_TOWARDZERO   3

typedef unsigned long int fexcept_t;
```

```
typedef unsigned long int fenv_t;

#define FE_DFL_ENV      ((__const fenv_t *) 0xc009804c0270033fUL)
```

10.6.3 math.h

```
typedef float float_t;
typedef double double_t;

#define fpclassify(x) \
    (sizeof (x) == sizeof (float) ? __fpclassifyf (x) : sizeof \
    (x) == sizeof (double) ? __fpclassify (x) : __fpclassifyl (x))
/* Return number of classification appropriate for X. */
#define signbit(x) \
    (sizeof (x) == sizeof (float)? __signbitf (x): sizeof (x) == \
    sizeof (double)? __signbit (x) : __signbitl (x)) /* Return nonzero \
    value if sign of X is negative. */
#define isfinite(x) \
    (sizeof (x) == sizeof (float) ? __finitef (x) : sizeof (x) == \
    sizeof (double)? __finite (x) : __finitel (x)) /* Return \
    nonzero value if X is not +-Inf or NaN. */
#define isinf(x) \
    (sizeof (x) == sizeof (float) ? __isinff (x): sizeof (x) == \
    sizeof (double) ? __isinf (x) : __isinfl (x))
#define isnan(x) \
    (sizeof (x) == sizeof (float) ? __isnanf (x) : sizeof (x) == \
    sizeof (double) ? __isnan (x) : __isnanl (x))

#define HUGE_VALL      0x1.0p32767L

#define FP_ILOGB0      -2147483648
#define FP_ILOGBNAN    2147483647

extern int __fpclassifyl(long double);
extern int __signbitl(long double);
extern long double exp2l(long double);
```

10.7 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 10.5 shall behave as described in the referenced base document. For interfaces referencing LSB and not listed below, please see the generic part of the specification.

__fpclassify**Name**

`__fpclassify` — Classify real floating type

Synopsis

```
int __fpclassify(long 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 long double.

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

__signbitl**Name**

`__signbitl` — test sign of floating point value

Synopsis

```
#include <math.h>
int __signbitl(long double arg);
```

Description

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

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

10.8 Interfaces for libpthread

Table 10-40 defines the library name and shared object name for the libpthread library

Table 10-40 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] LSB Core - Generic
- [SUSv3] POSIX 1003.1-2001 (ISO/IEC 9945-2003)
- [SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

10.8.1 Realtime Threads

10.8.1.1 Interfaces for Realtime Threads

An LSB conforming implementation shall provide the architecture specific functions for Realtime Threads specified in Table 10-41, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-41 libpthread - Realtime Threads Function Interfaces

pthread_attr_getinheritsched(GLIBC_2.2) [SUSv4]	pthread_attr_getschedpolicy(GLIBC_2.2) [SUSv4]	pthread_attr_getscope(GLIBC_2.2) [SUSv4]	pthread_attr_setinheritsched(GLIBC_2.2) [SUSv4]
pthread_attr_setschedpolicy(GLIBC_2.2) [SUSv4]	pthread_attr_setscope(GLIBC_2.2) [SUSv4]	pthread_getschedparam(GLIBC_2.2) [SUSv4]	pthread_setschedparam(GLIBC_2.2) [SUSv4]

10.8.2 Advanced Realtime Threads

10.8.2.1 Interfaces for Advanced Realtime Threads

An LSB conforming implementation shall provide the architecture specific functions for Advanced Realtime Threads specified in Table 10-42, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-42 libpthread - Advanced Realtime Threads Function Interfaces

pthread_barrier_destroy(GLIBC_2.2) [SUSv4]	pthread_barrier_init(GLIBC_2.2) [SUSv4]	pthread_barrier_wait(GLIBC_2.2) [SUSv4]	pthread_barrierattr_destroy(GLIBC_2.2) [SUSv4]
pthread_barrierattr_init(GLIBC_2.2) [SUSv4]	pthread_barrierattr_setshared(GLIBC_2.2) [SUSv4]	pthread_getcpuclockid(GLIBC_2.2) [SUSv4]	pthread_spin_destroy(GLIBC_2.2) [SUSv4]
pthread_spin_init(GLIBC_2.2) [SUSv4]	pthread_spin_lock(GLIBC_2.2) [SUSv4]	pthread_spin_trylock(GLIBC_2.2) [SUSv4]	pthread_spin_unlock(GLIBC_2.2) [SUSv4]

10.8.3 Posix Threads

10.8.3.1 Interfaces for Posix Threads

An LSB conforming implementation shall provide the architecture specific functions for Posix Threads specified in Table 10-43, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-43 libpthread - Posix Threads Function Interfaces

_pthread_cleanup_pop(GLIBC_2.2) [LSB]	_pthread_cleanup_push(GLIBC_2.2) [LSB]	pthread_attr_destroy(GLIBC_2.2) [SUSv4]	pthread_attr_getdetachstate(GLIBC_2.2) [SUSv4]
pthread_attr_getguardsize(GLIBC_2.2) [SUSv4]	pthread_attr_getschedparam(GLIBC_2.2) [SUSv4]	pthread_attr_getstack(GLIBC_2.2) [SUSv4]	pthread_attr_getstackaddr(GLIBC_2.2) [SUSv3]

pthread_attr_getstacksize(GLIBC_2.2) [SUSv4]	pthread_attr_init(GLIBC_2.2) [SUSv4]	pthread_attr_setdetachstate(GLIBC_2.2) [SUSv4]	pthread_attr_setguardsize(GLIBC_2.2) [SUSv4]
pthread_attr_setschedparam(GLIBC_2.2) [SUSv4]	pthread_attr_setstack(GLIBC_2.3.3) [SUSv4]	pthread_attr_setstackaddr(GLIBC_2.2) [SUSv3]	pthread_attr_setstacksize(GLIBC_2.3.3) [SUSv4]
pthread_cancel(GLIBC_2.2) [SUSv4]	pthread_cond_broadcast(GLIBC_2.3.2) [SUSv4]	pthread_cond_destroy(GLIBC_2.3.2) [SUSv4]	pthread_cond_init(GLIBC_2.3.2) [SUSv4]
pthread_cond_signal(GLIBC_2.3.2) [SUSv4]	pthread_cond_timedwait(GLIBC_2.3.2) [SUSv4]	pthread_cond_wait(GLIBC_2.3.2) [SUSv4]	pthread_condattr_destroy(GLIBC_2.2) [SUSv4]
pthread_condattr_getpshared(GLIBC_2.2) [SUSv4]	pthread_condattr_init(GLIBC_2.2) [SUSv4]	pthread_condattr_setpshared(GLIBC_2.2) [SUSv4]	pthread_create(GLIBC_2.2) [SUSv4]
pthread_detach(GLIBC_2.2) [SUSv4]	pthread_equal(GLIBC_2.2) [SUSv4]	pthread_exit(GLIBC_2.2) [SUSv4]	pthread_getconcurrency(GLIBC_2.2) [SUSv4]
pthread_getspecific(GLIBC_2.2) [SUSv4]	pthread_join(GLIBC_2.2) [SUSv4]	pthread_key_create(GLIBC_2.2) [SUSv4]	pthread_key_delete(GLIBC_2.2) [SUSv4]
pthread_kill(GLIBC_2.2) [SUSv4]	pthread_mutex_destroy(GLIBC_2.2) [SUSv4]	pthread_mutex_init(GLIBC_2.2) [SUSv4]	pthread_mutex_lock(GLIBC_2.2) [SUSv4]
pthread_mutex_timedlock(GLIBC_2.2) [SUSv4]	pthread_mutex_trylock(GLIBC_2.2) [SUSv4]	pthread_mutex_unlock(GLIBC_2.2) [SUSv4]	pthread_mutexattr_destroy(GLIBC_2.2) [SUSv4]
pthread_mutexattr_getpshared(GLIBC_2.2) [SUSv4]	pthread_mutexattr_gettype(GLIBC_2.2) [SUSv4]	pthread_mutexattr_init(GLIBC_2.2) [SUSv4]	pthread_mutexattr_setpshared(GLIBC_2.2) [SUSv4]
pthread_mutexattr_settype(GLIBC_2.2) [SUSv4]	pthread_once(GLIBC_2.2) [SUSv4]	pthread_rwlock_destroy(GLIBC_2.2) [SUSv4]	pthread_rwlock_init(GLIBC_2.2) [SUSv4]
pthread_rwlock_rdlock(GLIBC_2.2) [SUSv4]	pthread_rwlock_timedrdlock(GLIBC_2.2) [SUSv4]	pthread_rwlock_timedwrlock(GLIBC_2.2) [SUSv4]	pthread_rwlock_tryrdlock(GLIBC_2.2) [SUSv4]
pthread_rwlock_trywrlock(GLIBC_2.2) [SUSv4]	pthread_rwlock_unlock(GLIBC_2.2) [SUSv4]	pthread_rwlock_wrlock(GLIBC_2.2) [SUSv4]	pthread_rwlockattr_destroy(GLIBC_2.2) [SUSv4]
pthread_rwlockattr_getpshared(GLIBC_2.2) [SUSv4]	pthread_rwlockattr_init(GLIBC_2.2) [SUSv4]	pthread_rwlockattr_setpshared(GLIBC_2.2) [SUSv4]	pthread_self(GLIBC_2.2) [SUSv4]

IEC NORM.COM - Click to view the full PDF of ISO/IEC 23360-3-2:2021

pthread_setcancelstate(GLIBC_2.2) [SUSv4]	pthread_setcanceltype(GLIBC_2.2) [SUSv4]	pthread_setconcurrency(GLIBC_2.2) [SUSv4]	pthread_setspecific(GLIBC_2.2) [SUSv4]
pthread_sigmask(GLIBC_2.2) [SUSv4]	pthread_testcancel(GLIBC_2.2) [SUSv4]	sem_close(GLIBC_2.2) [SUSv4]	sem_destroy(GLIBC_2.2) [SUSv4]
sem_getvalue(GLIBC_2.2) [SUSv4]	sem_init(GLIBC_2.2) [SUSv4]	sem_open(GLIBC_2.2) [SUSv4]	sem_post(GLIBC_2.2) [SUSv4]
sem_timedwait(GLIBC_2.2) [SUSv4]	sem_trywait(GLIBC_2.2) [SUSv4]	sem_unlink(GLIBC_2.2) [SUSv4]	sem_wait(GLIBC_2.2) [SUSv4]

An LSB conforming implementation shall provide the architecture specific deprecated functions for Posix Threads specified in Table 10-44, 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 10-44 libpthread - Posix Threads Deprecated Function Interfaces

pthread_attr_getstackaddr(GLIBC_2.2) [SUSv3]	pthread_attr_setstackaddr(GLIBC_2.2) [SUSv3]		
--	--	--	--

10.8.4 Thread aware versions of libc interfaces

10.8.4.1 Interfaces for Thread aware versions of libc interfaces

An LSB conforming implementation shall provide the architecture specific functions for Thread aware versions of libc interfaces specified in Table 10-45, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-45 libpthread - Thread aware versions of libc interfaces Function Interfaces

lseek64(GLIBC_2.2) [LFS]	open64(GLIBC_2.2) [LFS]	pread(GLIBC_2.2) [SUSv4]	pread64(GLIBC_2.2) [LSB]
pwrite(GLIBC_2.2) [SUSv4]	pwrite64(GLIBC_2.2) [LSB]		

10.8.5 GNU Extensions for libpthread

10.8.5.1 Interfaces for GNU Extensions for libpthread

An LSB conforming implementation shall provide the architecture specific functions for GNU Extensions for libpthread specified in Table 10-46, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-46 libpthread - GNU Extensions for libpthread Function Interfaces

pthread_getattr_np(GLIBC_2.2.3) [LSB]	pthread_mutex_consistent_np(GLIBC_2.4) [LSB]	pthread_mutexattr_getrobust_np(GLIBC_2.4) [LSB]	pthread_mutexattr_setrobust_np(GLIBC_2.4) [LSB]
pthread_rwlockattr_getkind_np(GLIBC_2.2) [LSB]	pthread_rwlockattr_setkind_np(GLIBC_2.2) [LSB]		

10.8.6 System Calls

10.8.6.1 Interfaces for System Calls

An LSB conforming implementation shall provide the architecture specific functions for System Calls specified in Table 10-47, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-47 libpthread - System Calls Function Interfaces

close(GLIBC_2.2) [SUSv4]	fcntl(GLIBC_2.2) [LSB]	fork(GLIBC_2.2) [SUSv4]	fsync(GLIBC_2.2) [SUSv4]
lseek(GLIBC_2.2) [SUSv4]	msync(GLIBC_2.2) [SUSv4]	nanosleep(GLIBC_2.2) [SUSv4]	open(GLIBC_2.2) [SUSv4]
pause(GLIBC_2.2) [SUSv4]	read(GLIBC_2.2) [SUSv4]	vfork(GLIBC_2.2) [SUSv3]	wait(GLIBC_2.2) [SUSv4]
waitpid(GLIBC_2.2) [LSB]	write(GLIBC_2.2) [SUSv4]		

10.8.7 Standard I/O

10.8.7.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the architecture specific functions for Standard I/O specified in Table 10-48, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-48 libpthread - Standard I/O Function Interfaces

flockfile(GLIBC_2.2) [SUSv4]			
------------------------------	--	--	--

10.8.8 Signal Handling

10.8.8.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the architecture specific functions for Signal Handling specified in Table 10-49, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-49 libpthread - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.2) [LSB]	__libc_current_sigrtmin(GLIBC_2.2) [LSB]	raise(GLIBC_2.2) [SUSv4]	sigaction(GLIBC_2.2) [SUSv4]
--	--	--------------------------	------------------------------

siglongjmp(GLIB C_2.2) [SUSv4]	sigwait(GLIBC_2 .2) [SUSv4]		
-----------------------------------	--------------------------------	--	--

10.8.9 Standard Library

10.8.9.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the architecture specific functions for Standard Library specified in Table 10-50, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-50 libpthread - Standard Library Function Interfaces

__errno_location(GLIBC_2.2) [LSB]	ftrylockfile(GLIB C_2.2) [SUSv4]	funlockfile(GLIB C_2.2) [SUSv4]	longjmp(GLIBC_2 .2) [SUSv4]
system(GLIBC_2. 2) [LSB]			

10.8.10 Socket Interface

10.8.10.1 Interfaces for Socket Interface

An LSB conforming implementation shall provide the architecture specific functions for Socket Interface specified in Table 10-51, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-51 libpthread - Socket Interface Function Interfaces

__h_errno_locati on(GLIBC_2.2) [LSB]	accept(GLIBC_2 .2) [SUSv4]	connect(GLIBC_2 .2) [SUSv4]	recv(GLIBC_2.2) [SUSv4]
recvfrom(GLIBC _2.2) [SUSv4]	recvmsg(GLIBC_ 2.2) [SUSv4]	send(GLIBC_2.2) [SUSv4]	sendmsg(GLIBC _2.2) [SUSv4]
sendto(GLIBC_2. 2) [SUSv4]			

10.8.11 Terminal Interface Functions

10.8.11.1 Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the architecture specific functions for Terminal Interface Functions specified in Table 10-52, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-52 libpthread - Terminal Interface Functions Function Interfaces

tcdrain(GLIBC_2. 2) [SUSv4]			
--------------------------------	--	--	--

10.9 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.

10.9.1 lsb/pthread.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.9.2 pthread.h

```
#define __SIZEOF_PTHREAD_BARRIER_T 32
#define __SIZEOF_PTHREAD_MUTEX_T 40
#define __SIZEOF_PTHREAD_ATTR_T 56
#define __SIZEOF_PTHREAD_RWLOCK_T 56
#define PTHREAD_RWLOCK_INITIALIZER { { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } }
#define PTHREAD_MUTEX_INITIALIZER { { 0, 0, 0, 0, 0, 0, 0, { 0, 0 } } }

typedef union {
    char __size[__SIZEOF_PTHREAD_BARRIER_T];
    long int __align;
} pthread_barrier_t;

typedef struct __pthread_internal_list __pthread_list_t;
struct __pthread_mutex_s {
    int __lock;
    unsigned int __count;
    int __owner;
    unsigned int __nusers;
    int __kind;
    int __spins;
    __pthread_list_t __list;
};

typedef union {
    struct {
        int __lock;
        unsigned int __nr_readers;
        unsigned int __readers_wakeup;
        unsigned int __writer_wakeup;
        unsigned int __nr_readers_queued;
        unsigned int __nr_writers_queued;
        int __writer;
        int __pad1;
        unsigned long int __pad2;
        unsigned long int __pad3;
        unsigned int __flags;
    } __data;
    char __size[__SIZEOF_PTHREAD_RWLOCK_T];
};
```

```

    long int __align;
} pthread_rwlock_t;

```

10.9.3 semaphore.h

```

#define __SIZEOF_SEM_T 32

```

10.10 Interfaces for libgcc_s

Table 10-53 defines the library name and shared object name for the libgcc_s library

Table 10-53 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] LSB Core - Generic

10.10.1 Unwind Library

10.10.1.1 Interfaces for Unwind Library

An LSB conforming implementation shall provide the architecture specific functions for Unwind Library specified in Table 10-54, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-54 libgcc_s - Unwind Library Function Interfaces

_Unwind_Backtrace(GCC_3.3) [LSB]	_Unwind_DeleteException(GCC_3.0) [LSB]	_Unwind_FindEnclosingFunction(GCC_3.3) [LSB]	_Unwind_ForceUnwind(GCC_3.0) [LSB]
_Unwind_GetBSP(GCC_3.3.2) [LSB]	_Unwind_GetCFA(GCC_3.3) [LSB]	_Unwind_GetGR(GCC_3.0) [LSB]	_Unwind_GetIP(GCC_3.0) [LSB]
_Unwind_GetLanguageSpecificData(GCC_3.0) [LSB]	_Unwind_GetRegionStart(GCC_3.0) [LSB]	_Unwind_RaiseException(GCC_3.0) [LSB]	_Unwind_Resume(GCC_3.0) [LSB]
_Unwind_Resume_or_Rethrow(GCC_3.3) [LSB]	_Unwind_SetGR(GCC_3.0) [LSB]	_Unwind_SetIP(GCC_3.0) [LSB]	

10.11 Data Definitions for libgcc_s

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

10.11.1 unwind.h

```
extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);
```

10.12 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 10.10 shall behave as described in the referenced base document. For interfaces referencing LSB and not listed below, please see the generic part of the specification.

_Unwind_GetBSP

Name

_Unwind_GetBSP — private C++ error handling method

Synopsis

```
_Unwind_Word _Unwind_GetBSP(struct _Unwind_Context * context);
```

Description

_Unwind_GetBSP() shall retrieve the value of the Backing Store Pointer (BSP) of the given context.

10.13 Interfaces for libdl

Table 10-55 defines the library name and shared object name for the libdl library

Table 10-55 libdl Definition

Library:	libdl
SONAME:	libdl.so.2

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

[LSB] LSB Core - Generic

[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

10.13.1 Dynamic Loader

10.13.1.1 Interfaces for Dynamic Loader

An LSB conforming implementation shall provide the architecture specific functions for Dynamic Loader specified in Table 10-56, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-56 libdl - Dynamic Loader Function Interfaces

dladdr(GLIBC_2.0) [LSB]	dlclose(GLIBC_2.0) [SUSv4]	dlderror(GLIBC_2.0) [SUSv4]	dlopen(GLIBC_2.1) [LSB]
dlvsym(GLIBC_2.0) [LSB]	dlvsym(GLIBC_2.1) [LSB]		

10.14 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.

10.14.1 dlfcn.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

10.15 Interfaces for libcrypt

Table 10-57 defines the library name and shared object name for the libcrypt library

Table 10-57 libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

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

[LSB] LSB Core - Generic
[SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)

10.15.1 Encryption

10.15.1.1 Interfaces for Encryption

An LSB conforming implementation shall provide the architecture specific functions for Encryption specified in Table 10-58, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-58 libcrypt - Encryption Function Interfaces

crypt(GLIBC_2.0) [SUSv4]	crypt_r(GLIBC_2.0) [LSB]	encrypt(GLIBC_2.0) [SUSv4]	encrypt_r(GLIBC_2.0) [LSB]
setkey(GLIBC_2.0) [SUSv4]	setkey_r(GLIBC_2.0) [LSB]		

10.16 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.

10.16.1 crypt.h

```

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

```

IV Utility Libraries

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

11 Libraries

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.

11.1 Interfaces for libz

Table 11-1 defines the library name and shared object name for the libz library

Table 11-1 libz Definition

Library:	libz
SONAME:	libz.so.1

11.1.1 Compression Library

11.1.1.1 Interfaces for Compression Library

No external functions are defined for libz - Compression Library in this part of the specification. See also the generic specification.

11.2 Data Definitions for libz

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

11.2.1 zconf.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.2.2 zlib.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.3 Interfaces for libncurses

Table 11-2 defines the library name and shared object name for the libncurses library

Table 11-2 libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

11.3.1 Curses

11.3.1.1 Interfaces for Curses

No external functions are defined for libncurses - Curses in this part of the specification. See also the generic specification.

11.4 Data Definitions for libncurses

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

11.4.1 curses.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.5 Interfaces for libncursesw

Table 11-3 defines the library name and shared object name for the libncursesw library

Table 11-3 libncursesw Definition

Library:	libncursesw
SONAME:	libncursesw.so.5

11.5.1 Curses Wide

11.5.1.1 Interfaces for Curses Wide

No external functions are defined for libncursesw - Curses Wide in this part of the specification. See also the generic specification.

11.6 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.

11.6.1 ncursesw/curses.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.6.2 ncursesw/ncurses_dll.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.6.3 ncursesw/term.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.6.4 ncursesw/unctrl.h

```
/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */
```

11.7 Interfaces for libutil

Table 11-4 defines the library name and shared object name for the libutil library

Table 11-4 libutil Definition

Library:	libutil
SONAME:	libutil.so.1

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

[LSB] LSB Core - Generic

11.7.1 Utility Functions

11.7.1.1 Interfaces for Utility Functions

An LSB conforming implementation shall provide the architecture specific functions for Utility Functions specified in Table 11-5, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-5 libutil - Utility Functions Function Interfaces

forkpty(GLIBC_2.0) [LSB]	login(GLIBC_2.0) [LSB]	login_tty(GLIBC_2.0) [LSB]	logout(GLIBC_2.0) [LSB]
logwtmp(GLIBC_2.0) [LSB]	openpty(GLIBC_2.0) [LSB]		

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

V Base Libraries

IECNORM.COM : Click to view the full PDF of ISO/IEC 23360-3-2:2021

12 Libraries

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

Only those interfaces that are unique to the Itanium™ platform are defined here. This section should be used in conjunction with the corresponding section in the Linux Standard Base Specification.

12.1 Interfaces for libstdcxx

Table 12-1 defines the library name and shared object name for the libstdcxx library

Table 12-1 libstdcxx Definition

Library:	libstdcxx
SONAME:	libstdc++.so.6

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] LSB Core - Generic

12.1.1 C++ Runtime Support

12.1.1.1 Interfaces for C++ Runtime Support

An LSB conforming implementation shall provide the architecture specific methods for C++ Runtime Support specified in Table 12-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-2 libstdcxx - C++ Runtime Support Function Interfaces

operator new[](unsigned long)(GLIBCXX_3.4) [ISOCXX]
operator new[](unsigned long, nothrow_t const&)(GLIBCXX_3.4) [ISOCXX]
operator new(unsigned long)(GLIBCXX_3.4) [ISOCXX]
operator new(unsigned long, nothrow_t const&)(GLIBCXX_3.4) [ISOCXX]

12.1.2 C++ type descriptors for built-in types

12.1.2.1 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 generic specification.

12.1.3 C++ _Rb_tree

12.1.3.1 Interfaces for C++ _Rb_tree

No external methods are defined for libstdcxx - C++ _Rb_tree in this part of the specification. See also the generic specification.

12.1.4 Class type_info

12.1.4.1 Class data for type_info

The virtual table for the `std::type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `std::type_info` class is described by Table 12-3

Table 12-3 typeinfo for type_info

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for <code>type_info</code>

12.1.4.2 Interfaces for Class type_info

No external methods are defined for `libstdc++` - Class `std::type_info` in this part of the specification. See also the generic specification.

12.1.5 Class __cxxabiv1::__enum_type_info

12.1.5.1 Class data for __cxxabiv1::__enum_type_info

The virtual table for the `__cxxabiv1::__enum_type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `__cxxabiv1::__enum_type_info` class is described by Table 12-4

Table 12-4 typeinfo for __cxxabiv1::__enum_type_info

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__enum_type_info</code>

12.1.5.2 Interfaces for Class __cxxabiv1::__enum_type_info

No external methods are defined for `libstdc++` - Class `__cxxabiv1::__enum_type_info` in this part of the specification. See also the generic specification.

12.1.6 Class __cxxabiv1::__array_type_info

12.1.6.1 Class data for __cxxabiv1::__array_type_info

The virtual table for the `__cxxabiv1::__array_type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `__cxxabiv1::__array_type_info` class is described by Table 12-5

Table 12-5 typeinfo for __cxxabiv1::__array_type_info

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
-------------	---

Name	typeinfo name for __cxxabiv1::__array_type_info
------	--

12.1.6.2 Interfaces for Class __cxxabiv1::__array_type_info

No external methods are defined for libstdc++ - Class __cxxabiv1::__array_type_info in this part of the specification. See also the generic specification.

12.1.7 Class __cxxabiv1::__class_type_info

12.1.7.1 Class data for __cxxabiv1::__class_type_info

The virtual table for the __cxxabiv1::__class_type_info class is described by Table 12-6

Table 12-6 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
vfunc[7]:	__cxxabiv1::__class_type_info::__do_dyn_cast(long, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyn_cast_result&) const

vfunc[8]:	__cxxabiv1::__class_type_info::__do_find_public_src(long, void const*, __cxxabiv1::__class_type_info const*, void const*) const
-----------	---

The Run Time Type Information for the __cxxabiv1::__class_type_info class is described by Table 12-7

Table 12-7 typeinfo for __cxxabiv1::__class_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__class_type_info

12.1.7.2 Interfaces for Class __cxxabiv1::__class_type_info

An LSB conforming implementation shall provide the architecture specific methods for Class __cxxabiv1::__class_type_info specified in Table 12-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-8 libstdcxx - Class __cxxabiv1::__class_type_info Function Interfaces

__cxxabiv1::__class_type_info::__do_dyncast(long, __cxxabiv1::__class_type_info::__sub_kind __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyncast_result&) const(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__class_type_info::__do_find_public_src(long, void const*, __cxxabiv1::__class_type_info const*, void const*) const(CXXABI_1.3) [CXXABI-1.86]

12.1.8 Class __cxxabiv1::__pbase_type_info

12.1.8.1 Class data for __cxxabiv1::__pbase_type_info

The virtual table for the __cxxabiv1::__pbase_type_info class is described in the generic part of this specification.

The Run Time Type Information for the __cxxabiv1::__pbase_type_info class is described by Table 12-9

Table 12-9 typeinfo for __cxxabiv1::__pbase_type_info

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for __cxxabiv1::__pbase_type_info

12.1.8.2 Interfaces for Class __cxxabiv1::__pbase_type_info

No external methods are defined for libstdcxx - Class __cxxabiv1::__pbase_type_info in this part of the specification. See also the generic specification.

12.1.9 Class `__cxxabiv1::__pointer_type_info`

12.1.9.1 Class data for `__cxxabiv1::__pointer_type_info`

The virtual table for the `__cxxabiv1::__pointer_type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `__cxxabiv1::__pointer_type_info` class is described by Table 12-10

Table 12-10 typeinfo for `__cxxabiv1::__pointer_type_info`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__pointer_type_info</code>

12.1.9.2 Interfaces for Class `__cxxabiv1::__pointer_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::__pointer_type_info` in this part of the specification. See also the generic specification.

12.1.10 Class `__cxxabiv1::__function_type_info`

12.1.10.1 Class data for `__cxxabiv1::__function_type_info`

The virtual table for the `__cxxabiv1::__function_type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `__cxxabiv1::__function_type_info` class is described by Table 12-11

Table 12-11 typeinfo for `__cxxabiv1::__function_type_info`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__function_type_info</code>

12.1.10.2 Interfaces for Class `__cxxabiv1::__function_type_info`

No external methods are defined for `libstdcxx` - Class `__cxxabiv1::__function_type_info` in this part of the specification. See also the generic specification.

12.1.11 Class `__cxxabiv1::__si_class_type_info`

12.1.11.1 Class data for `__cxxabiv1::__si_class_type_info`

The virtual table for the `__cxxabiv1::__si_class_type_info` class is described by Table 12-12

Table 12-12 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::__d o_upcast(__cxxabiv1::__class_type_in fo const*, void const*, __cxxabiv1::__class_type_info::__upc ast_result&) const
vfunc[7]:	__cxxabiv1::__si_class_type_info::__d o_dyn_cast(long, __cxxabiv1::__class_type_info::__sub _kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyn cast_result&) const
vfunc[8]:	__cxxabiv1::__si_class_type_info::__d o_find_public_src(long, void const*, __cxxabiv1::__class_type_info const*, void const*) const

The Run Time Type Information for the __cxxabiv1::__si_class_type_info class is described by Table 12-13

Table 12-13 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

12.1.11.2 Interfaces for Class __cxxabiv1::__si_class_type_info

An LSB conforming implementation shall provide the architecture specific methods for Class __cxxabiv1::__si_class_type_info specified in Table 12-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-14 libstdc++ - Class __cxxabiv1::__si_class_type_info Function Interfaces

__cxxabiv1::__si_class_type_info::__do_dynccast(long, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dynccast_result&) const(CXXABI_1.3) [CXXABI-1.86]
__cxxabiv1::__si_class_type_info::__do_find_public_src(long, void const*, __cxxabiv1::__class_type_info const*, void const*) const(CXXABI_1.3) [CXXABI-1.86]

12.1.12 Class __cxxabiv1::__vmi_class_type_info

12.1.12.1 Class data for __cxxabiv1::__vmi_class_type_info

The virtual table for the __cxxabiv1::__vmi_class_type_info class is described by Table 12-15

Table 12-15 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::__upc ast_result&) const

vfunc[7]:	<code>__cxxabiv1::__vmi_class_type_info:: _do_dyncast(long, __cxxabiv1::__class_type_info::__sub _kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyn cast_result&) const</code>
vfunc[8]:	<code>__cxxabiv1::__vmi_class_type_info:: _do_find_public_src(long, void const*, __cxxabiv1::__class_type_info const*, void const*) const</code>

The Run Time Type Information for the `__cxxabiv1::__vmi_class_type_info` class is described by Table 12-16

Table 12-16 typeinfo for `__cxxabiv1::__vmi_class_type_info`

Base Vtable	vtable for <code>__cxxabiv1::__vmi_class_type_info</code>
Name	typeinfo name for <code>__cxxabiv1::__vmi_class_type_info</code>

12.1.12.2 Interfaces for Class `__cxxabiv1::__vmi_class_type_info`

An LSB conforming implementation shall provide the architecture specific methods for Class `__cxxabiv1::__vmi_class_type_info` specified in Table 12-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-17 libstdc++ - Class `__cxxabiv1::__vmi_class_type_info` Function Interfaces

<code>__cxxabiv1::__vmi_class_type_info::_do_dyncast(long, __cxxabiv1::__class_type_info::__sub_kind, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info const*, void const*, __cxxabiv1::__class_type_info::__dyncast_result&) const(CXXABI_1.3) [CXXABI-1.86]</code>
<code>__cxxabiv1::__vmi_class_type_info::_do_find_public_src(long, void const*, __cxxabiv1::__class_type_info const*, void const*) const(CXXABI_1.3) [CXXABI-1.86]</code>

12.1.13 Class `__cxxabiv1::__fundamental_type_info`

12.1.13.1 Class data for `__cxxabiv1::__fundamental_type_info`

The virtual table for the `__cxxabiv1::__fundamental_type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `__cxxabiv1::__fundamental_type_info` class is described by Table 12-18

Table 12-18 typeid for `__cxxabiv1::__fundamental_type_info`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>__cxxabiv1::__fundamental_type_info</code>

12.1.13.2 Interfaces for Class**`__cxxabiv1::__fundamental_type_info`**

No external methods are defined for `libstdc++` - Class `__cxxabiv1::__fundamental_type_info` in this part of the specification. See also the generic specification.

12.1.14 Class**`__cxxabiv1::__pointer_to_member_type_info`****12.1.14.1 Class data for****`__cxxabiv1::__pointer_to_member_type_info`**

The virtual table for the `__cxxabiv1::__pointer_to_member_type_info` class is described in the generic part of this specification.

The Run Time Type Information for the `__cxxabiv1::__pointer_to_member_type_info` class is described by Table 12-19

Table 12-19 typeid for `__cxxabiv1::__pointer_to_member_type_info`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>__cxxabiv1::__pointer_to_member_type_info</code>

12.1.14.2 Interfaces for Class**`__cxxabiv1::__pointer_to_member_type_info`**

No external methods are defined for `libstdc++` - Class `__cxxabiv1::__pointer_to_member_type_info` in this part of the specification. See also the generic specification.

12.1.15 Class `__gnu_cxx::stdio_filebuf<char, char_traits<char> >`**12.1.15.1 Interfaces for Class `__gnu_cxx::stdio_filebuf<char, char_traits<char> >`**

No external methods are defined for `libstdc++` - Class `__gnu_cxx::stdio_filebuf<char, std::char_traits<char> >` in this part of the specification. See also the generic specification.

12.1.16 Class `__gnu_cxx::stdio_filebuf<wchar_t, char_traits<wchar_t> >`

12.1.16.1 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 generic specification.

12.1.17 Class `__gnu_cxx::__pool_alloc_base`

12.1.17.1 Interfaces for Class `__gnu_cxx::__pool_alloc_base`

An LSB conforming implementation shall provide the architecture specific methods for Class `__gnu_cxx::__pool_alloc_base` specified in Table 12-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-20 `libstdcxx` - Class `__gnu_cxx::__pool_alloc_base` Function Interfaces

<code>__gnu_cxx::__pool_alloc_base::_M_get_free_list(unsigned long)(GLIBCXX_3.4.2) [LSB]</code>
<code>__gnu_cxx::__pool_alloc_base::_M_refill(unsigned long)(GLIBCXX_3.4.2) [LSB]</code>

12.1.18 Class `__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >`

12.1.18.1 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 12-21

Table 12-21 Primary vtable for `__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >`

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeid</code> for <code>__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> ></code>
<code>vfunc[0]:</code>	<code>__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::~stdio_sync_filebuf()</code>
<code>vfunc[1]:</code>	<code>__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::~stdio_sync_filebuf()</code>
<code>vfunc[2]:</code>	<code>basic_streambuf<char, char_traits<char> >::imbue(locale const&)</code>

vfunc[3]:	basic_streambuf<char, char_traits<char> >::setbuf(char*, long)
vfunc[4]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::xsgetn(char*, long)
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]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::xspn(char const*, long)
vfunc[13]:	__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >::overflow(int)

12.1.18.2 Interfaces for Class

__gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >

No external methods are defined for libstdc++ - Class __gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char> > in this part of the specification. See also the generic specification.

12.1.19 Class __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >

12.1.19.1 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 12-22

Table 12-22 _cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >

Base Offset	0
-------------	---

Virtual Base Offset	0
RTTI	typeid for __gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >
vfunc[0]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::~stdio_sync_filebuf()
vfunc[1]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::~stdio_sync_filebuf()
vfunc[2]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::imbue(locale const&)
vfunc[3]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, long)
vfunc[4]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
vfunc[5]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::seekpos(fpos<__mbstate_t>, _Ios_Openmode)
vfunc[6]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::sync()
vfunc[7]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::showmanyc()
vfunc[8]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::xsgetn(wchar_t*, long)
vfunc[9]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::underflow()
vfunc[10]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::uflow()
vfunc[11]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::pbackfail(unsigned int)
vfunc[12]:	__gnu_cxx::stdio_sync_filebuf<wcha r_t, char_traits<wchar_t> >::xspn(wchar_t const*, long)

IECNORM.COM : Click to view the full PDF of ISO 23360-3-2:2021

vfunc[13]:	__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t>>::overflow(unsigned int)
------------	--

12.1.19.2 Interfaces for Class

__gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t>>

No external methods are defined for libstdc++ - Class __gnu_cxx::stdio_sync_filebuf<wchar_t, std::char_traits<wchar_t>> in this part of the specification. See also the generic specification.

12.1.20 Class exception

12.1.20.1 Class data for exception

The virtual table for the std::exception class is described in the generic part of this specification.

The Run Time Type Information for the std::exception class is described by Table 12-23

Table 12-23 typeinfo for exception

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeid name for exception

12.1.20.2 Interfaces for Class exception

No external methods are defined for libstdc++ - Class std::exception in this part of the specification. See also the generic specification.

12.1.21 Class bad_typeid

12.1.21.1 Class data for bad_typeid

The virtual table for the std::bad_typeid class is described in the generic part of this specification.

The Run Time Type Information for the std::bad_typeid class is described by Table 12-24

Table 12-24 typeinfo for bad_typeid

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for bad_typeid

12.1.21.2 Interfaces for Class bad_typeid

No external methods are defined for libstdc++ - Class std::bad_typeid in this part of the specification. See also the generic specification.

12.1.22 Class `logic_error`

12.1.22.1 Class data for `logic_error`

The virtual table for the `std::logic_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::logic_error` class is described by Table 12-25

Table 12-25 typeinfo for `logic_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>logic_error</code>

12.1.22.2 Interfaces for Class `logic_error`

No external methods are defined for `libstdc++` - Class `std::logic_error` in this part of the specification. See also the generic specification.

12.1.23 Class `range_error`

12.1.23.1 Class data for `range_error`

The virtual table for the `std::range_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::range_error` class is described by Table 12-26

Table 12-26 typeinfo for `range_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>range_error</code>

12.1.23.2 Interfaces for Class `range_error`

No external methods are defined for `libstdc++` - Class `std::range_error` in this part of the specification. See also the generic specification.

12.1.24 Class `domain_error`

12.1.24.1 Class data for `domain_error`

The virtual table for the `std::domain_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::domain_error` class is described by Table 12-27

Table 12-27 typeinfo for `domain_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>domain_error</code>

12.1.24.2 Interfaces for Class `domain_error`

No external methods are defined for `libstdc++` - Class `std::domain_error` in this part of the specification. See also the generic specification.

12.1.25 Class `length_error`

12.1.25.1 Class data for `length_error`

The virtual table for the `std::length_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::length_error` class is described by Table 12-28

Table 12-28 `typeinfo` for `length_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>length_error</code>

12.1.25.2 Interfaces for Class `length_error`

No external methods are defined for `libstdc++` - Class `std::length_error` in this part of the specification. See also the generic specification.

12.1.26 Class `out_of_range`

12.1.26.1 Class data for `out_of_range`

The virtual table for the `std::out_of_range` class is described in the generic part of this specification.

The Run Time Type Information for the `std::out_of_range` class is described by Table 12-29

Table 12-29 `typeinfo` for `out_of_range`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>out_of_range</code>

12.1.26.2 Interfaces for Class `out_of_range`

No external methods are defined for `libstdc++` - Class `std::out_of_range` in this part of the specification. See also the generic specification.

12.1.27 Class `bad_exception`

12.1.27.1 Class data for `bad_exception`

The virtual table for the `std::bad_exception` class is described in the generic part of this specification.

The Run Time Type Information for the `std::bad_exception` class is described by Table 12-30

Table 12-30 typeinfo for `bad_exception`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>bad_exception</code>

12.1.27.2 Interfaces for Class `bad_exception`

No external methods are defined for `libstdc++` - Class `std::bad_exception` in this part of the specification. See also the generic specification.

12.1.28 Class `runtime_error`

12.1.28.1 Class data for `runtime_error`

The virtual table for the `std::runtime_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::runtime_error` class is described by Table 12-31

Table 12-31 typeinfo for `runtime_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>runtime_error</code>

12.1.28.2 Interfaces for Class `runtime_error`

No external methods are defined for `libstdc++` - Class `std::runtime_error` in this part of the specification. See also the generic specification.

12.1.29 Class `overflow_error`

12.1.29.1 Class data for `overflow_error`

The virtual table for the `std::overflow_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::overflow_error` class is described by Table 12-32

Table 12-32 typeinfo for `overflow_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>overflow_error</code>

12.1.29.2 Interfaces for Class `overflow_error`

No external methods are defined for `libstdc++` - Class `std::overflow_error` in this part of the specification. See also the generic specification.

12.1.30 Class `underflow_error`

12.1.30.1 Class data for `underflow_error`

The virtual table for the `std::underflow_error` class is described in the generic part of this specification.

The Run Time Type Information for the `std::underflow_error` class is described by Table 12-33

Table 12-33 typeinfo for `underflow_error`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>underflow_error</code>

12.1.30.2 Interfaces for Class `underflow_error`

No external methods are defined for `libstdc++` - Class `std::underflow_error` in this part of the specification. See also the generic specification.

12.1.31 Class `invalid_argument`

12.1.31.1 Class data for `invalid_argument`

The virtual table for the `std::invalid_argument` class is described in the generic part of this specification.

The Run Time Type Information for the `std::invalid_argument` class is described by Table 12-34

Table 12-34 typeinfo for `invalid_argument`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>invalid_argument</code>

12.1.31.2 Interfaces for Class `invalid_argument`

No external methods are defined for `libstdc++` - Class `std::invalid_argument` in this part of the specification. See also the generic specification.

12.1.32 Class `bad_cast`

12.1.32.1 Class data for `bad_cast`

The virtual table for the `std::bad_cast` class is described in the generic part of this specification.

The Run Time Type Information for the `std::bad_cast` class is described by Table 12-35

Table 12-35 typeinfo for `bad_cast`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>bad_cast</code>

12.1.32.2 Interfaces for Class `bad_cast`

No external methods are defined for `libstdc++` - Class `std::bad_cast` in this part of the specification. See also the generic specification.

12.1.33 Class `bad_alloc`

12.1.33.1 Class data for `bad_alloc`

The virtual table for the `std::bad_alloc` class is described in the generic part of this specification.

The Run Time Type Information for the `std::bad_alloc` class is described by Table 12-36

Table 12-36 `typeinfo` for `bad_alloc`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>bad_alloc</code>

12.1.33.2 Interfaces for Class `bad_alloc`

No external methods are defined for `libstdc++` - Class `std::bad_alloc` in this part of the specification. See also the generic specification.

12.1.34 struct `__numeric_limits_base`

12.1.34.1 Interfaces for struct `__numeric_limits_base`

No external methods are defined for `libstdc++` - struct `__numeric_limits_base` in this part of the specification. See also the generic specification.

12.1.35 struct `numeric_limits<long double>`

12.1.35.1 Interfaces for struct `numeric_limits<long double>`

No external methods are defined for `libstdc++` - struct `numeric_limits<long double>` in this part of the specification. See also the generic specification.

12.1.36 struct `numeric_limits<long long>`

12.1.36.1 Interfaces for struct `numeric_limits<long long>`

No external methods are defined for `libstdc++` - struct `numeric_limits<long long>` in this part of the specification. See also the generic specification.

12.1.37 struct `numeric_limits<unsigned long long>`

12.1.37.1 Interfaces for struct `numeric_limits<unsigned long long>`

No external methods are defined for `libstdc++` - struct `numeric_limits<unsigned long long>` in this part of the specification. See also the generic specification.

12.1.38 struct numeric_limits<float>**12.1.38.1 Interfaces for struct numeric_limits<float>**

No external methods are defined for libstdc++ - struct numeric_limits<float> in this part of the specification. See also the generic specification.

12.1.39 struct numeric_limits<double>**12.1.39.1 Interfaces for struct numeric_limits<double>**

No external methods are defined for libstdc++ - struct numeric_limits<double> in this part of the specification. See also the generic specification.

12.1.40 struct numeric_limits<short>**12.1.40.1 Interfaces for struct numeric_limits<short>**

No external methods are defined for libstdc++ - struct numeric_limits<short> in this part of the specification. See also the generic specification.

12.1.41 struct numeric_limits<unsigned short>**12.1.41.1 Interfaces for struct numeric_limits<unsigned short>**

No external methods are defined for libstdc++ - struct numeric_limits<unsigned short> in this part of the specification. See also the generic specification.

12.1.42 struct numeric_limits<int>**12.1.42.1 Interfaces for struct numeric_limits<int>**

No external methods are defined for libstdc++ - struct numeric_limits<int> in this part of the specification. See also the generic specification.

12.1.43 struct numeric_limits<unsigned int>**12.1.43.1 Interfaces for struct numeric_limits<unsigned int>**

No external methods are defined for libstdc++ - struct numeric_limits<unsigned int> in this part of the specification. See also the generic specification.

12.1.44 struct numeric_limits<long>**12.1.44.1 Interfaces for struct numeric_limits<long>**

No external methods are defined for libstdc++ - struct numeric_limits<long> in this part of the specification. See also the generic specification.

12.1.45 struct numeric_limits<unsigned long>**12.1.45.1 Interfaces for struct numeric_limits<unsigned long>**

No external methods are defined for libstdc++ - struct numeric_limits<unsigned long> in this part of the specification. See also the generic specification.

12.1.46 struct numeric_limits<wchar_t>**12.1.46.1 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 generic specification.

12.1.47 struct numeric_limits<unsigned char>**12.1.47.1 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 generic specification.

12.1.48 struct numeric_limits<signed char>**12.1.48.1 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 generic specification.

12.1.49 struct numeric_limits<char>**12.1.49.1 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 generic specification.

12.1.50 struct numeric_limits<bool>**12.1.50.1 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 generic specification.

12.1.51 Class ctype_base**12.1.51.1 Class data for ctype_base**

The Run Time Type Information for the std::ctype_base class is described by Table 12-37.

Table 12-37 typeid for ctype_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeid name for ctype_base

12.1.51.2 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 generic specification.

IEC NORM.COM: Click to view the full PDF of ISO/IEC 23360-3-2:2021

12.1.52 Class `__ctype_abstract_base<char>`**12.1.52.1 Class data for `__ctype_abstract_base<char>`**

The virtual table for the `std::__ctype_abstract_base<char>` class is described in the generic part of this specification.

12.1.52.2 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 generic specification.

12.1.53 Class `__ctype_abstract_base<wchar_t>`**12.1.53.1 Class data for `__ctype_abstract_base<wchar_t>`**

The virtual table for the `std::__ctype_abstract_base<wchar_t>` class is described in the generic part of this specification.

12.1.53.2 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 generic specification.

12.1.54 Class `ctype<char>`**12.1.54.1 Class data for `ctype<char>`**

The virtual table for the `std::ctype<char>` class is described in the generic part of this specification.

12.1.54.2 Interfaces for Class `ctype<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::ctype<char>` specified in Table 12-38, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-38 `libstdcxx` - Class `ctype<char>` Function Interfaces

<code>ctype<char>::ctype(__locale_struct*, unsigned short const*, bool, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype<char>::ctype(unsigned short const*, bool, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype<char>::ctype(__locale_struct*, unsigned short const*, bool, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>ctype<char>::ctype(unsigned short const*, bool, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.55 Class `ctype<wchar_t>`

12.1.55.1 Class data for `ctype<wchar_t>`

The virtual table for the `std::ctype<wchar_t>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::ctype<wchar_t>` class is described by Table 12-39

Table 12-39 `typeinfo` for `ctype<wchar_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>ctype<wchar_t></code>

12.1.55.2 Interfaces for Class `ctype<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::ctype<wchar_t>` specified in Table 12-40, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-40 `libstdcxx` - Class `ctype<wchar_t>` Function Interfaces

<code>ctype<wchar_t>::ctype(__locale_struct*, unsigned long)(GLIBCXX_3.4)</code> [ISOCXX]
<code>ctype<wchar_t>::ctype(unsigned long)(GLIBCXX_3.4)</code> [ISOCXX]
<code>ctype<wchar_t>::ctype(__locale_struct*, unsigned long)(GLIBCXX_3.4)</code> [ISOCXX]
<code>ctype<wchar_t>::ctype(unsigned long)(GLIBCXX_3.4)</code> [ISOCXX]

12.1.56 Class `ctype_byname<char>`

12.1.56.1 Class data for `ctype_byname<char>`

The virtual table for the `std::ctype_byname<char>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::ctype_byname<char>` class is described by Table 12-41

Table 12-41 `typeinfo` for `ctype_byname<char>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>ctype_byname<char></code>

12.1.56.2 Interfaces for Class `ctype_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::ctype_byname<char>` specified in Table 12-42, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-42 libstdcxx - Class ctype_byname<char> Function Interfaces

ctype_byname<char>::ctype_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
ctype_byname<char>::ctype_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.57 Class ctype_byname<wchar_t>**12.1.57.1 Class data for ctype_byname<wchar_t>**

The virtual table for the std::ctype_byname<wchar_t> class is described in the generic part of this specification.

The Run Time Type Information for the std::ctype_byname<wchar_t> class is described by Table 12-43

Table 12-43 typeidinfo for ctype_byname<wchar_t>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeidinfo name for ctype_byname<wchar_t>

12.1.57.2 Interfaces for Class ctype_byname<wchar_t>

An LSB conforming implementation shall provide the architecture specific methods for Class std::ctype_byname<wchar_t> specified in Table 12-44, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-44 libstdcxx - Class ctype_byname<wchar_t> Function Interfaces

ctype_byname<wchar_t>::ctype_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
ctype_byname<wchar_t>::ctype_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.58 Class basic_string<char, char_traits<char>, allocator<char> >**12.1.58.1 Interfaces for Class basic_string<char, char_traits<char>, allocator<char> >**

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_string<char, std::char_traits<char>, std::allocator<char> > specified in Table 12-45, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-45 libstdcxx - Class basic_string<char, char_traits<char>, allocator<char> > Function Interfaces

basic_string<char, char_traits<char>, allocator<char> >::find_last_of(char const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]

<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_of(char const*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_of(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_of(char, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_of(char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_of(char const*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_of(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_of(char, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::M_check_length(unsigned long, unsigned long, char const*) const</code> (GLIBCXX_3.4.5) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_not_of(char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_not_of(char const*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_not_of(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_last_not_of(char, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_not_of(char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_not_of(char const*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_not_of(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find_first_not_of(char, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::at(unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::copy(char*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]

<code>basic_string<char, char_traits<char>, allocator<char> >::find(char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find(char const*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::find(char, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::rfind(char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::rfind(char const*, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::rfind(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::rfind(char, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::substr(unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::compare(unsigned long, unsigned long, char const*) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::compare(unsigned long, unsigned long, char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::compare(unsigned long, unsigned long, basic_string<char, char_traits<char>, allocator<char> > const&) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::compare(unsigned long, unsigned long, basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_check(unsigned long, char const*) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_limit(unsigned long, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::operator[](unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::_S_construct(unsigned long, char, allocator<char> const&)</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_replace_aux(unsigned long, unsigned long, unsigned long, char)</code> (GLIBCXX_3.4) [ISOCXX]

basic_string<char, char_traits<char>, allocator<char> >::M_replace_safe(unsigned long, unsigned long, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::at(unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::Rep::M_set_length_and_sharable(unsigned long)(GLIBCXX_3.4.5) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::Rep::M_clone(allocator<char> const&, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::Rep::S_create(unsigned long, unsigned long, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::erase(unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::append(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::append(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::append(unsigned long, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::assign(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::assign(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::assign(unsigned long, char)(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> >, unsigned long, char)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::insert(unsigned long, char const*)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::insert(unsigned long, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::insert(unsigned long, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]
basic_string<char, char_traits<char>, allocator<char> >::insert(unsigned long, basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]

<code>basic_string<char, char_traits<char>, allocator<char> >::insert(unsigned long, unsigned long, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::resize(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::resize(unsigned long, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_copy(char*, char const*, unsigned long)(GLIBCXX_3.4.5) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_move(char*, char const*, unsigned long)(GLIBCXX_3.4.5) [ISOCXX]</code>
<code>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*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>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> >, unsigned long, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::replace(unsigned long, unsigned long, char const*)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::replace(unsigned long, unsigned long, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::replace(unsigned long, unsigned long, basic_string<char, char_traits<char>, allocator<char> > const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::replace(unsigned long, unsigned long, basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::replace(unsigned long, unsigned long, unsigned long, char)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::reserve(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_assign(char*, unsigned long, char)(GLIBCXX_3.4.5) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::_M_mutate(unsigned long, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(char const*, unsigned long, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

<code>basic_string<char, char_traits<char>, allocator<char>> >::basic_string(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(unsigned long, char, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(char const*, unsigned long, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(basic_string<char, char_traits<char>, allocator<char> > const&, unsigned long, unsigned long, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::basic_string(unsigned long, char, allocator<char> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<char, char_traits<char>, allocator<char> >::operator[](unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.59 Class `basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

12.1.59.1 Interfaces for Class `basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` specified in Table 12-46, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-46. `libstdcxx` - Class `basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >` Function Interfaces

<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::find_last_of(wchar_t const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::find_last_of(wchar_t const*, unsigned long, unsigned long) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::find_last_of(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, unsigned long) const(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::find_last_of(wchar_t, unsigned long) const(GLIBCXX_3.4) [ISOCXX]</code>

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_of(wchar_t const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_of(wchar_t const*, unsigned long, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_of(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_of(wchar_t, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::M_check_length(unsigned long, unsigned long, char const*) const(GLIBCXX_3.4.5) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_last_not_of(wchar_t const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_last_not_of(wchar_t const*, unsigned long, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_last_not_of(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_last_not_of(wchar_t, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_not_of(wchar_t const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_not_of(wchar_t const*, unsigned long, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_not_of(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find_first_not_of(wchar_t, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::at(unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::copy(wchar_t*, unsigned long, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find(wchar_t const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]

basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find(wchar_t const*, unsigned long, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::find(wchar_t, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::rfind(wchar_t const*, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::rfind(wchar_t const*, unsigned long, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::rfind(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::rfind(wchar_t, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::substr(unsigned long, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::compare(unsigned long, unsigned long, wchar_t const*) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::compare(unsigned long, unsigned long, wchar_t const*, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::compare(unsigned long, unsigned long, 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>>::compare(unsigned long, unsigned long, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::_M_check(unsigned long, char const*) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::_M_limit(unsigned long, unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::operator[](unsigned long) const (GLIBCXX_3.4) [ISOCXX]
basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::_S_construct(unsigned long, wchar_t, allocator<wchar_t> const&) (GLIBCXX_3.4) [ISOCXX]

<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::M_replace_aux(unsigned long, unsigned long, unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::M_replace_safe(unsigned long, unsigned long, wchar_t const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::at(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::Rep::M_set_length_and_sharable(unsigned long)(GLIBCXX_3.4.5) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::Rep::M_clone(allocator<wchar_t> const&, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::Rep::S_create(unsigned long, unsigned long, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::erase(unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::append(wchar_t const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::append(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::append(unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::assign(wchar_t const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::assign(basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::assign(unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>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>>>, unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::insert(unsigned long, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::insert(unsigned long, wchar_t const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::insert(unsigned long, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> const&)(GLIBCXX_3.4) [ISOCXX]</code>

<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::insert(unsigned long, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::insert(unsigned long, unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::resize(unsigned long)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::resize(unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::M_copy(wchar_t*, wchar_t const*, unsigned long)(GLIBCXX_3.4.5) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::M_move(wchar_t*, wchar_t const*, unsigned long)(GLIBCXX_3.4.5) [ISOCXX]</pre>
<pre>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*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>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> > >, unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(unsigned long, unsigned long, wchar_t const*)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(unsigned long, unsigned long, wchar_t const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(unsigned long, unsigned long, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(unsigned long, unsigned long, basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > const&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</pre>
<pre>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::replace(unsigned long, unsigned long, unsigned long, wchar_t)(GLIBCXX_3.4) [ISOCXX]</pre>

<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::reserve(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::_M_assign(wchar_t*, unsigned long, wchar_t)(GLIBCXX_3.4.5) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::_M_mutate(unsigned long, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::basic_string(wchar_t const*, unsigned long, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>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&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>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&, unsigned long, unsigned long, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::basic_string(unsigned long, wchar_t, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::basic_string(wchar_t const*, unsigned long, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>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&, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>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&, unsigned long, unsigned long, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::basic_string(unsigned long, wchar_t, allocator<wchar_t> const&)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::operator[](unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.60 Class `basic_stringstream<char, char_traits<char>, allocator<char> >`

12.1.60.1 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 by Table 12-47

Table 12-47 Primary vtable for `basic_stringstream<char, char_traits<char>, allocator<char> >`

Base Offset	0
Virtual Base Offset	104
RTTI	typeinfo for <code>basic_stringstream<char, char_traits<char>, allocator<char> ></code>
vfunc[0]:	<code>basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()</code>
vfunc[1]:	<code>basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()</code>

Table 12-48 Secondary vtable for `basic_stringstream<char, char_traits<char>, allocator<char> >`

Base Offset	-16
Virtual Base Offset	88
RTTI	typeinfo for <code>basic_stringstream<char, char_traits<char>, allocator<char> ></code>
vfunc[0]:	non-virtual thunk to <code>basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()</code>
vfunc[1]:	non-virtual thunk to <code>basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()</code>

Table 12-49 Secondary vtable for `basic_stringstream<char, char_traits<char>, allocator<char> >`

Base Offset	-104
Virtual Base Offset	-104
RTTI	typeinfo for <code>basic_stringstream<char, char_traits<char>, allocator<char> ></code>
vfunc[0]:	virtual thunk to <code>basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()</code>
vfunc[1]:	virtual thunk to <code>basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()</code>

The VTT for the `std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 12-50

Table 12-50 VTT for basic_stringstream<char, char_traits<char>, allocator<char> >

VTT Name	_ZTTSt18basic_stringstreamlcSt11char_traitslcESaIcEE
Number of Entries	10

12.1.60.2 Interfaces for Class basic_stringstream<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_stringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 12-51, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-51 libstdcxx - Class basic_stringstream<char, char_traits<char>, allocator<char> > Function Interfaces

non-virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [CXXABI-1.86]
non-virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_stringstream<char, char_traits<char>, allocator<char> >::~basic_stringstream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.61 Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

12.1.61.1 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 by Table 12-52

Table 12-52 Primary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Offset	0
Virtual Base Offset	104
RTTI	typeinfo for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()

vfunc[1]:	basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()
-----------	--

Table 12-53 Secondary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Offset	-16
Virtual Base Offset	88
RTTI	typeid for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	non-virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()
vfunc[1]:	non-virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()

Table 12-54 Secondary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Offset	-104
Virtual Base Offset	-104
RTTI	typeid for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()
vfunc[1]:	virtual thunk to basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()

The VTT for the std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 12-55

Table 12-55 VTT for `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

VTT Name	<code>_ZTTSt18basic_stringstreamlwSt11char_traitslwESaIwEE</code>
Number of Entries	10

12.1.61.2 Interfaces for Class `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_stringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` specified in Table 12-56, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-56 `libstdcxx` - Class `basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >` Function Interfaces

non-virtual thunk to <code>basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]
non-virtual thunk to <code>basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_stringstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]

12.1.62 Class `basic_istream<char, char_traits<char>, allocator<char> >`

12.1.62.1 Class data for `basic_istream<char, char_traits<char>, allocator<char> >`

The virtual table for the `std::basic_istream<char, std::char_traits<char>, std::allocator<char> >` class is described by Table 12-57

Table 12-57 Primary vtable for `basic_istream<char, char_traits<char>, allocator<char> >`

Base Offset	0
Virtual Base Offset	96
RTTI	typeinfo for <code>basic_istream<char, char_traits<char>, allocator<char> ></code>
<code>vfunc[0]:</code>	<code>basic_istream<char, char_traits<char>, allocator<char> >::~basic_istream()</code>
<code>vfunc[1]:</code>	<code>basic_istream<char, char_traits<char>, allocator<char> >::~basic_istream()</code>

Table 12-58 Secondary vtable for basic_istream<char, char_traits<char>, allocator<char> >

Base Offset	-96
Virtual Base Offset	-96
RTTI	typeinfo for basic_istream<char, char_traits<char>, allocator<char> >
vfunc[0]:	virtual thunk to basic_istream<char, char_traits<char>, allocator<char> >::~basic_istream()
vfunc[1]:	virtual thunk to basic_istream<char, char_traits<char>, allocator<char> >::~basic_istream()

The VTT for the std::basic_istream<char, std::char_traits<char>, std::allocator<char> > class is described by Table 12-59

Table 12-59 VTT for basic_istream<char, char_traits<char>, allocator<char> >

VTT Name	_ZTISt19basic_istreamIcSt11char_traitsIcESaIcEE
Number of Entries	4

12.1.62.2 Interfaces for Class basic_istream<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_istream<char, std::char_traits<char>, std::allocator<char> > specified in Table 12-60, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-60 libstdc++ - Class basic_istream<char, char_traits<char>, allocator<char> > Function Interfaces

virtual thunk to basic_istream<char, char_traits<char>, allocator<char> >::~basic_istream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_istream<char, char_traits<char>, allocator<char> >::~basic_istream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.63 Class basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

12.1.63.1 Class data for basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

The virtual table for the std::basic_istream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 12-61

Table 12-61 Primary vtable for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

Base Offset	0
Virtual Base Offset	96
RTTI	typeinfo for <code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>></code>
<code>vfunc[0]:</code>	<code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_istream()</code>
<code>vfunc[1]:</code>	<code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_istream()</code>

Table 12-62 Secondary vtable for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

Base Offset	-96
Virtual Base Offset	-96
RTTI	typeinfo for <code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>></code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_istream()</code>
<code>vfunc[1]:</code>	virtual thunk to <code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_istream()</code>

The VTT for the `std::basic_istream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>` class is described by Table 12-63

Table 12-63 VTT for `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

VTT Name	<code>_ZTTSt19basic_istreamIwSt11char_traitsIwESaIwEE</code>
Number of Entries	4

12.1.63.2 Interfaces for Class `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_istream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>>` specified in Table 12-64, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-64 `libstdcxx` - Class `basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>` Function Interfaces

virtual thunk to <code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_istream()(GLIBCXX_3.4)</code> [CXXABI-1.86]
virtual thunk to <code>basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_istream()(GLIBCXX_3.4)</code> [CXXABI-1.86]

12.1.64 Class `basic_ostringstream<char, char_traits<char>, allocator<char>>`

12.1.64.1 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 by Table 12-65

Table 12-65 Primary vtable for `basic_ostringstream<char, char_traits<char>, allocator<char>>`

Base Offset	0
Virtual Base Offset	88
RTTI	typeinfo for <code>basic_ostringstream<char, char_traits<char>, allocator<char>></code>
<code>vfunc[0]:</code>	<code>basic_ostringstream<char, char_traits<char>, allocator<char>>::~basic_ostringstream()</code>
<code>vfunc[1]:</code>	<code>basic_ostringstream<char, char_traits<char>, allocator<char>>::~basic_ostringstream()</code>

Table 12-66 Secondary vtable for `basic_ostringstream<char, char_traits<char>, allocator<char>>`

Base Offset	-88
Virtual Base Offset	-88
RTTI	typeinfo for <code>basic_ostringstream<char, char_traits<char>, allocator<char>></code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_ostringstream<char,</code>

	char_traits<char>, allocator<char> >::~basic_ostringstream()
vfunc[1]:	virtual thunk to basic_ostringstream<char, char_traits<char>, allocator<char> >::~basic_ostringstream()

The VTT for the std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> > class is described by Table 12-67

Table 12-67 VTT for basic_ostringstream<char, char_traits<char>, allocator<char> >

VTT Name	_ZTTSt19basic_ostringstreamIcSt11c har_traitsIcESaIcEE
Number of Entries	4

12.1.64.2 Interfaces for Class basic_ostringstream<char, char_traits<char>, allocator<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_ostringstream<char, std::char_traits<char>, std::allocator<char> > specified in Table 12-68, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-68 libstdc++ - Class basic_ostringstream<char, char_traits<char>, allocator<char> > Function Interfaces

virtual thunk to basic_ostringstream<char, char_traits<char>, allocator<char> >::~basic_ostringstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_ostringstream<char, char_traits<char>, allocator<char> >::~basic_ostringstream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.65 Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

12.1.65.1 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 by Table 12-69

Table 12-69 Primary vtable for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Offset	0
Virtual Base Offset	88
RTTI	typeinfo for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

vfunc[0]:	basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()
vfunc[1]:	basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()

Table 12-70 Secondary vtable for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

Base Offset	-88
Virtual Base Offset	-88
RTTI	typeinfo for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >
vfunc[0]:	virtual thunk to basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()
vfunc[1]:	virtual thunk to basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::~basic_ostringstream()

The VTT for the std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > class is described by Table 12-71

Table 12-71 VTT for basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

VTT Name	_ZTTSt19basic_ostringstreamIwSt11char_traitsIwESaIwEE
Number of Entries	4

12.1.65.2 Interfaces for Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_ostringstream<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > specified in Table 12-72, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-72 libstdcxx - Class basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> Function Interfaces

virtual thunk to basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_ostringstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_ostringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>::~basic_ostringstream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.66 Class basic_stringbuf<char, char_traits<char>, allocator<char>>

12.1.66.1 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 12-73

Table 12-73 Primary vtable for basic_stringbuf<char, char_traits<char>, allocator<char>>

Base Offset	0
Virtual Base Offset	0
RTTI	typeid 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]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::setbuf(char*, long)
vfunc[4]:	basic_stringbuf<char, char_traits<char>, allocator<char>>::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	basic_streambuf<char, char_traits<char> >::xsgetn(char*, long)
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]:	basic_streambuf<char, char_traits<char> >::xspn(char, const*, long)
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 12-74

Table 12-74 typeinfo for `basic_stringbuf<char, char_traits<char>, allocator<char> >`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>basic_stringbuf<char, char_traits<char>, allocator<char> ></code>

12.1.66.2 Interfaces for Class `basic_stringbuf<char, char_traits<char>, allocator<char> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_stringbuf<char, std::char_traits<char>, std::allocator<char> >` specified in Table 12-75, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-75 libstdc++ - Class `basic_stringbuf<char, char_traits<char>, allocator<char> >` Function Interfaces

<code>basic_stringbuf<char, char_traits<char>, allocator<char> >::setbuf(char*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_stringbuf<char, char_traits<char>, allocator<char> >::_M_sync(char*, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_stringbuf<char, char_traits<char>, allocator<char> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.67 Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

12.1.67.1 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 12-76

Table 12-76 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]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::setbuf(wchar_t*, long)
vfunc[4]:	basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, long)
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]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, long)
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 12-77

Table 12-77 typeid for `basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >

12.1.67.2 Interfaces for Class `basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_stringbuf<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >` specified in Table 12-78, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-78 libstdc++ - Class `basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >` Function Interfaces

<code>basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::setbuf(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >::_M_sync(wchar_t*, unsigned long, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

```
basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t>
>::seekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
```

12.1.68 Class basic_iostream<char, char_traits<char> >

12.1.68.1 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 by Table 12-79

Table 12-79 Primary vtable for basic_iostream<char, char_traits<char> >

Base Offset	0
Virtual Base Offset	24
RTTI	typeinfo for basic_iostream<char, char_traits<char> >
vfunc[0]:	basic_iostream<char, char_traits<char> >::~basic_iostream()
vfunc[1]:	basic_iostream<char, char_traits<char> >::~basic_iostream()

Table 12-80 Secondary vtable for basic_iostream<char, char_traits<char> >

Base Offset	-16
Virtual Base Offset	8
RTTI	typeinfo for basic_iostream<char, char_traits<char> >
vfunc[0]:	non-virtual thunk to basic_iostream<char, char_traits<char> >::~basic_iostream()
vfunc[1]:	non-virtual thunk to basic_iostream<char, char_traits<char> >::~basic_iostream()

Table 12-81 Secondary vtable for basic_iostream<char, char_traits<char> >

Base Offset	-24
Virtual Base Offset	-24
RTTI	typeinfo for basic_iostream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_iostream<char, char_traits<char> >::~basic_iostream()

vfunc[1]:	virtual thunk to basic_iostream<char, char_traits<char>> >::~basic_iostream()
-----------	--

The VTT for the `std::basic_iostream<char, std::char_traits<char>>` class is described by Table 12-82

Table 12-82 VTT for `basic_iostream<char, char_traits<char>>`

VTT Name	_ZTTSd
Number of Entries	7

12.1.68.2 Interfaces for Class `basic_iostream<char, char_traits<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_iostream<char, std::char_traits<char>>` specified in Table 12-83, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-83 libstdc++ - Class `basic_iostream<char, char_traits<char>>` Function Interfaces

non-virtual thunk to <code>basic_iostream<char, char_traits<char>></code> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]
non-virtual thunk to <code>basic_iostream<char, char_traits<char>></code> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_iostream<char, char_traits<char>></code> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_iostream<char, char_traits<char>></code> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.69 Class `basic_iostream<wchar_t, char_traits<wchar_t>>`

12.1.69.1 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 by Table 12-84

Table 12-84 Primary vtable for `basic_iostream<wchar_t, char_traits<wchar_t>>`

Base Offset	0
Virtual Base Offset	24
RTTI	typeinfo for <code>basic_iostream<wchar_t, char_traits<wchar_t>></code>
vfunc[0]:	<code>basic_iostream<wchar_t, char_traits<wchar_t>></code> >::~basic_iostream()

vfunc[1]:	basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()
-----------	---

Table 12-85 Secondary vtable for basic_iostream<wchar_t, char_traits<wchar_t> >

Base Offset	-16
Virtual Base Offset	8
RTTI	typeinfo for basic_iostream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	non-virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()
vfunc[1]:	non-virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()

Table 12-86 Secondary vtable for basic_iostream<wchar_t, char_traits<wchar_t> >

Base Offset	24
Virtual Base Offset	-24
RTTI	typeinfo for basic_iostream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()
vfunc[1]:	virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()

The VTT for the std::basic_iostream<wchar_t, std::char_traits<wchar_t> > class is described by Table 12-87

Table 12-87 VTT for basic_iostream<wchar_t, char_traits<wchar_t> >

VTT Name	_ZTTSt14basic_iostreamlwSt11char_t raitslwEE
Number of Entries	7

12.1.69.2 Interfaces for Class basic_iostream<wchar_t, char_traits<wchar_t> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_iostream<wchar_t, std::char_traits<wchar_t> >

specified in Table 12-88, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-88 libstdcxx - Class basic_iostream<wchar_t, char_traits<wchar_t> > Function Interfaces

non-virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]
non-virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_iostream<wchar_t, char_traits<wchar_t> >::~basic_iostream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.70 Class basic_istream<char, char_traits<char> >

12.1.70.1 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 by Table 12-89

Table 12-89 Primary vtable for basic_istream<char, char_traits<char> >

Base Offset	0
Virtual Base Offset	16
RTTI	typeinfo for basic_istream<char, char_traits<char> >
vfunc[0]:	basic_istream<char, char_traits<char> >::~~basic_istream()
vfunc[1]:	basic_istream<char, char_traits<char> >::~~basic_istream()

Table 12-90 Secondary vtable for basic_istream<char, char_traits<char> >

Base Offset	-16
Virtual Base Offset	-16
RTTI	typeinfo for basic_istream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_istream<char, char_traits<char> >::~~basic_istream()
vfunc[1]:	virtual thunk to basic_istream<char, char_traits<char> >::~~basic_istream()

The VTT for the std::basic_istream<char, std::char_traits<char> > class is described by Table 12-91

Table 12-91 VTT for `basic_istream<char, char_traits<char>>`

VTT Name	<code>_ZTTSi</code>
Number of Entries	2

12.1.70.2 Interfaces for Class `basic_istream<char, char_traits<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_istream<char, std::char_traits<char>>` specified in Table 12-92, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-92 `libstdc++` - Class `basic_istream<char, char_traits<char>>` Function Interfaces

<code>basic_istream<char, char_traits<char>>::get(char*, long)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::get(char*, long, char)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::read(char*, long)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::seekg(long, _Ios_Seekdir)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::ignore(long)(GLIBCXX_3.4.5)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::ignore(long, int)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::getline(char*, long)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::getline(char*, long, char)(GLIBCXX_3.4)</code> [ISOCXX]
<code>basic_istream<char, char_traits<char>>::readsome(char*, long)(GLIBCXX_3.4)</code> [ISOCXX]
virtual thunk to <code>basic_istream<char, char_traits<char>>::~basic_istream()(GLIBCXX_3.4)</code> [CXXABI-1.86]
virtual thunk to <code>basic_istream<char, char_traits<char>>::~basic_istream()(GLIBCXX_3.4)</code> [CXXABI-1.86]

12.1.71 Class `basic_istream<wchar_t, char_traits<wchar_t>>`

12.1.71.1 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 by Table 12-93

Table 12-93 Primary vtable for `basic_istream<wchar_t, char_traits<wchar_t> >`

Base Offset	0
Virtual Base Offset	16
RTTI	typeinfo for <code>basic_istream<wchar_t, char_traits<wchar_t> ></code>
<code>vfunc[0]:</code>	<code>basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()</code>
<code>vfunc[1]:</code>	<code>basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()</code>

Table 12-94 Secondary vtable for `basic_istream<wchar_t, char_traits<wchar_t> >`

Base Offset	-16
Virtual Base Offset	-16
RTTI	typeinfo for <code>basic_istream<wchar_t, char_traits<wchar_t> ></code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()</code>
<code>vfunc[1]:</code>	virtual thunk to <code>basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()</code>

The VTT for the `std::basic_istream<wchar_t, std::char_traits<wchar_t> >` class is described by Table 12-95

Table 12-95 VTT for `basic_istream<wchar_t, char_traits<wchar_t> >`

VTT Name	<code>_ZTTSt13basic_istreamIwSt11char_traitsIwEE</code>
Number of Entries	2

12.1.71.2 Interfaces for Class `basic_istream<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_istream<wchar_t, std::char_traits<wchar_t> >` specified in Table 12-96, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-96 libstdcxx - Class basic_istream<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_istream<wchar_t, char_traits<wchar_t> >::get(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::get(wchar_t*, long, wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::read(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::seekg(long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::ignore(long)(GLIBCXX_3.4.5) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::ignore(long, unsigned int)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::getline(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::getline(wchar_t*, long, wchar_t)(GLIBCXX_3.4) [ISOCXX]
basic_istream<wchar_t, char_traits<wchar_t> >::readsome(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_istream<wchar_t, char_traits<wchar_t> >::~basic_istream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.72 Class istreambuf_iterator<wchar_t, char_traits<wchar_t> >

12.1.72.1 Interfaces for Class istreambuf_iterator<wchar_t, char_traits<wchar_t> >

No external methods are defined for libstdcxx - Class std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t> > in this part of the specification. See also the generic specification.

12.1.73 Class istreambuf_iterator<char, char_traits<char> >

12.1.73.1 Interfaces for Class istreambuf_iterator<char, char_traits<char> >

No external methods are defined for libstdcxx - Class std::istreambuf_iterator<char, std::char_traits<char> > in this part of the specification. See also the generic specification.

12.1.74 Class `basic_ostream<char, char_traits<char>>`

12.1.74.1 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 by Table 12-97

Table 12-97 Primary vtable for `basic_ostream<char, char_traits<char>>`

Base Offset	0
Virtual Base Offset	8
RTTI	<code>typeid for basic_ostream<char, char_traits<char>></code>
<code>vfunc[0]:</code>	<code>basic_ostream<char, char_traits<char>>::~~basic_ostream()</code>
<code>vfunc[1]:</code>	<code>basic_ostream<char, char_traits<char>>::~~basic_ostream()</code>

Table 12-98 Secondary vtable for `basic_ostream<char, char_traits<char>>`

Base Offset	-8
Virtual Base Offset	-8
RTTI	<code>typeid for basic_ostream<char, char_traits<char>></code>
<code>vfunc[0]:</code>	<code>virtual thunk to basic_ostream<char, char_traits<char>>::~~basic_ostream()</code>
<code>vfunc[1]:</code>	<code>virtual thunk to basic_ostream<char, char_traits<char>>::~~basic_ostream()</code>

The VTT for the `std::basic_ostream<char, std::char_traits<char>>` class is described by Table 12-99

Table 12-99 VTT for `basic_ostream<char, char_traits<char>>`

VTT Name	<code>_ZTTSo</code>
Number of Entries	2

12.1.74.2 Interfaces for Class `basic_ostream<char, char_traits<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ostream<char, std::char_traits<char>>` specified in Table 12-100, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-100 libstdcxx - Class basic_ostream<char, char_traits<char> > Function Interfaces

basic_ostream<char, char_traits<char> >::seekp(long, _Ios_Seekdir)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::write(char const*, long)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<char, char_traits<char> >::_M_write(char const*, long)(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_ostream<char, char_traits<char> >::~basic_ostream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.75 Class basic_ostream<wchar_t, char_traits<wchar_t> >

12.1.75.1 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 by Table 12-101

Table 12-101 Primary vtable for basic_ostream<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	8
RTTI	typeinfo for basic_ostream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()
vfunc[1]:	basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()

Table 12-102 Secondary vtable for basic_ostream<wchar_t, char_traits<wchar_t> >

Base Offset	-8
Virtual Base Offset	-8
RTTI	typeinfo for basic_ostream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()

vfunc[1]:	virtual thunk to basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()
-----------	---

The VTT for the `std::basic_ostream<wchar_t, std::char_traits<wchar_t> >` class is described by Table 12-103

Table 12-103 VTT for `basic_ostream<wchar_t, char_traits<wchar_t> >`

VTT Name	_ZTTSt13basic_ostreamIwSt11char_t raitsIwEE
Number of Entries	2

12.1.75.2 Interfaces for Class `basic_ostream<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ostream<wchar_t, std::char_traits<wchar_t> >` specified in Table 12-104, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-104 `libstdcxx` - Class `basic_ostream<wchar_t, char_traits<wchar_t> >` Function Interfaces

<code>basic_ostream<wchar_t, char_traits<wchar_t> >::seekp(long, _Ios_Seekdir)</code> (GLIBCXX_3.4) [ISOCXX]
<code>basic_ostream<wchar_t, char_traits<wchar_t> >::write(wchar_t const*, long)</code> (GLIBCXX_3.4) [ISOCXX]
virtual thunk to <code>basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()</code> (GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_ostream<wchar_t, char_traits<wchar_t> >::~basic_ostream()</code> (GLIBCXX_3.4) [CXXABI-1.86]

12.1.76 Class `basic_fstream<char, char_traits<char> >`

12.1.76.1 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 by Table 12-105

Table 12-105 Primary vtable for `basic_fstream<char, char_traits<char> >`

Base Offset	0
Virtual Base Offset	264
RTTI	<code>typeinfo for basic_fstream<char, char_traits<char> ></code>
vfunc[0]:	<code>basic_fstream<char, char_traits<char> >::~basic_fstream()</code>

vfunc[1]:	basic_fstream<char, char_traits<char> >::~~basic_fstream()
-----------	---

Table 12-106 Secondary vtable for basic_fstream<char, char_traits<char> >

Base Offset	-16
Virtual Base Offset	248
RTTI	typeinfo for basic_fstream<char, char_traits<char> >
vfunc[0]:	non-virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()
vfunc[1]:	non-virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()

Table 12-107 Secondary vtable for basic_fstream<char, char_traits<char> >

Base Offset	-264
Virtual Base Offset	-264
RTTI	typeinfo for basic_fstream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()
vfunc[1]:	virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()

The VTT for the std::basic_fstream<char, std::char_traits<char> > class is described by Table 12-108

Table 12-108 VTT for basic_fstream<char, char_traits<char> >

VTT Name	_ZTTSt13basic_fstreamIcSt11char_traitsIcEE
Number of Entries	10

12.1.76.2 Interfaces for Class basic_fstream<char, char_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_fstream<char, std::char_traits<char> > specified in Table 12-109, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-109 libstdcxx - Class basic_fstream<char, char_traits<char> > Function Interfaces

non-virtual thunk to basic_fstream<char, char_traits<char> >::~~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]
--

non-virtual thunk to basic_fstream<char, char_traits<char> >::~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_fstream<char, char_traits<char> >::~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_fstream<char, char_traits<char> >::~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.77 Class basic_fstream<wchar_t, char_traits<wchar_t> >

12.1.77.1 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 by Table 12-110

Table 12-110 Primary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	264
RTTI	typeid for basic_fstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()
vfunc[1]:	basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()

Table 12-111 Secondary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >

Base Offset	-16
Virtual Base Offset	248
RTTI	typeid for basic_fstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	non-virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()
vfunc[1]:	non-virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()

Table 12-112 Secondary vtable for `basic_fstream<wchar_t, char_traits<wchar_t>>`

Base Offset	-264
Virtual Base Offset	-264
RTTI	typeinfo for <code>basic_fstream<wchar_t, char_traits<wchar_t>></code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_fstream<wchar_t, char_traits<wchar_t>>::~~basic_fstream()</code>
<code>vfunc[1]:</code>	virtual thunk to <code>basic_fstream<wchar_t, char_traits<wchar_t>>::~~basic_fstream()</code>

The VTT for the `std::basic_fstream<wchar_t, std::char_traits<wchar_t>>` class is described by Table 12-113

Table 12-113 VTT for `basic_fstream<wchar_t, char_traits<wchar_t>>`

VTT Name	<code>_ZTTSt13basic_fstreamlwSt11char_traitslwEE</code>
Number of Entries	10

12.1.77.2 Interfaces for Class `basic_fstream<wchar_t, char_traits<wchar_t>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_fstream<wchar_t, std::char_traits<wchar_t>>` specified in Table 12-114, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-114 `libstdc++` - Class `basic_fstream<wchar_t, char_traits<wchar_t>>` Function Interfaces

non-virtual thunk to <code>basic_fstream<wchar_t, char_traits<wchar_t>>::~~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]</code>
non-virtual thunk to <code>basic_fstream<wchar_t, char_traits<wchar_t>>::~~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]</code>
virtual thunk to <code>basic_fstream<wchar_t, char_traits<wchar_t>>::~~basic_fstream()(GLIBCXX_3.4) [CXXABI-1.86]</code>

12.1.78 Class `basic_ifstream<char, char_traits<char>>`

12.1.78.1 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 by Table 12-115

Table 12-115 Primary vtable for basic_ifstream<char, char_traits<char> >

Base Offset	0
Virtual Base Offset	256
RTTI	typeinfo for basic_ifstream<char, char_traits<char> >
vfunc[0]:	basic_ifstream<char, char_traits<char> >::~basic_ifstream()
vfunc[1]:	basic_ifstream<char, char_traits<char> >::~basic_ifstream()

Table 12-116 Secondary vtable for basic_ifstream<char, char_traits<char> >

Base Offset	-256
Virtual Base Offset	-256
RTTI	typeinfo for basic_ifstream<char, char_traits<char> >
vfunc[0]:	virtual thunk to basic_ifstream<char, char_traits<char> >::~basic_ifstream()
vfunc[1]:	virtual thunk to basic_ifstream<char, char_traits<char> >::~basic_ifstream()

The VTT for the std::basic_ifstream<char, std::char_traits<char> > class is described by Table 12-117.

Table 12-117 VTT for basic_ifstream<char, char_traits<char> >

VTT Name	_ZTTSt14basic_ifstreamIcSt11char_traitsIcEE
Number of Entries	4

12.1.78.2 Interfaces for Class basic_ifstream<char, char_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_ifstream<char, std::char_traits<char> > specified in Table 12-118, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-118 libstdc++ - Class basic_ifstream<char, char_traits<char> > Function Interfaces

virtual thunk to basic_ifstream<char, char_traits<char> >::~basic_ifstream()(GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to basic_ifstream<char, char_traits<char> >::~basic_ifstream()(GLIBCXX_3.4) [CXXABI-1.86]

12.1.79 Class `basic_ifstream<wchar_t, char_traits<wchar_t> >`

12.1.79.1 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 by Table 12-119

Table 12-119 Primary vtable for `basic_ifstream<wchar_t, char_traits<wchar_t> >`

Base Offset	0
Virtual Base Offset	256
RTTI	typeinfo for <code>basic_ifstream<wchar_t, char_traits<wchar_t> ></code>
<code>vfunc[0]:</code>	<code>basic_ifstream<wchar_t, char_traits<wchar_t> >::~~basic_ifstream()</code>
<code>vfunc[1]:</code>	<code>basic_ifstream<wchar_t, char_traits<wchar_t> >::~~basic_ifstream()</code>

Table 12-120 Secondary vtable for `basic_ifstream<wchar_t, char_traits<wchar_t> >`

Base Offset	-256
Virtual Base Offset	-256
RTTI	typeinfo for <code>basic_ifstream<wchar_t, char_traits<wchar_t> ></code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_ifstream<wchar_t, char_traits<wchar_t> >::~~basic_ifstream()</code>
<code>vfunc[1]:</code>	virtual thunk to <code>basic_ifstream<wchar_t, char_traits<wchar_t> >::~~basic_ifstream()</code>

The VTT for the `std::basic_ifstream<wchar_t, std::char_traits<wchar_t> >` class is described by Table 12-121

Table 12-121 VTT for `basic_ifstream<wchar_t, char_traits<wchar_t> >`

VTT Name	<code>_ZTTSt14basic_ifstreamIwSt11char_traitsIwEE</code>
Number of Entries	4

12.1.79.2 Interfaces for Class `basic_ifstream<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ifstream<wchar_t, std::char_traits<wchar_t> >` specified in Table 12-122, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-122 libstdc++ - Class `basic_ifstream<wchar_t, char_traits<wchar_t> >` Function Interfaces

virtual thunk to <code>basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_ifstream<wchar_t, char_traits<wchar_t> >::~basic_ifstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]

12.1.80 Class `basic_ofstream<char, char_traits<char> >`

12.1.80.1 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 by Table 12-123

Table 12-123 Primary vtable for `basic_ofstream<char, char_traits<char> >`

Base Offset	0
Virtual Base Offset	248
RTTI	<code>typeinfo</code> for <code>basic_ofstream<char, char_traits<char> ></code>
<code>vfunc[0]:</code>	<code>basic_ofstream<char, char_traits<char> >::~basic_ofstream()</code>
<code>vfunc[1]:</code>	<code>basic_ofstream<char, char_traits<char> >::~basic_ofstream()</code>

Table 12-124 Secondary vtable for `basic_ofstream<char, char_traits<char> >`

Base Offset	-248
Virtual Base Offset	-248
RTTI	<code>typeinfo</code> for <code>basic_ofstream<char, char_traits<char> ></code>
<code>vfunc[0]:</code>	virtual thunk to <code>basic_ofstream<char, char_traits<char> >::~basic_ofstream()</code>
<code>vfunc[1]:</code>	virtual thunk to <code>basic_ofstream<char, char_traits<char> >::~basic_ofstream()</code>

The VTT for the `std::basic_ofstream<char, std::char_traits<char> >` class is described by Table 12-125

Table 12-125 VTT for `basic_ofstream<char, char_traits<char> >`

VTT Name	<code>_ZTTSt14basic_ofstreamIcSt11char_traitsIcEE</code>
Number of Entries	4

12.1.80.2 Interfaces for Class `basic_ofstream<char, char_traits<char> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ofstream<char, std::char_traits<char> >` specified in Table 12-126, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-126 `libstdcxx` - Class `basic_ofstream<char, char_traits<char> >` Function Interfaces

virtual thunk to <code>basic_ofstream<char, char_traits<char> >::~basic_ofstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]
virtual thunk to <code>basic_ofstream<char, char_traits<char> >::~basic_ofstream()</code> (GLIBCXX_3.4) [CXXABI-1.86]

12.1.81 Class `basic_ofstream<wchar_t, char_traits<wchar_t> >`

12.1.81.1 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 by Table 12-127

Table 12-127 Primary vtable for `basic_ofstream<wchar_t, char_traits<wchar_t> >`

Base Offset	0
Virtual Base Offset	248
RTTI	<code>typeinfo</code> for <code>basic_ofstream<wchar_t, char_traits<wchar_t> ></code>
<code>vfunc[0]:</code>	<code>basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()</code>
<code>vfunc[1]:</code>	<code>basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()</code>

Table 12-128 Secondary vtable for `basic_ofstream<wchar_t, char_traits<wchar_t> >`

Base Offset	-248
-------------	------

Virtual Base Offset	-248
RTTI	typeinfo for basic_ofstream<wchar_t, char_traits<wchar_t> >
vfunc[0]:	virtual thunk to basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()
vfunc[1]:	virtual thunk to basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()

The VTT for the `std::basic_ofstream<wchar_t, std::char_traits<wchar_t> >` class is described by Table 12-129

Table 12-129 VTT for `basic_ofstream<wchar_t, char_traits<wchar_t> >`

VTT Name	<code>_ZTTSt14basic_ofstreamlwSt11char_traitslwEE</code>
Number of Entries	4

12.1.81.2 Interfaces for Class `basic_ofstream<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_ofstream<wchar_t, std::char_traits<wchar_t> >` specified in Table 12-130, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-130 `libstdcxx` - Class `basic_ofstream<wchar_t, char_traits<wchar_t> >` Function Interfaces

virtual thunk to <code>basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()(GLIBCXX_3.4) [CXXABI-1.86]</code>
virtual thunk to <code>basic_ofstream<wchar_t, char_traits<wchar_t> >::~basic_ofstream()(GLIBCXX_3.4) [CXXABI-1.86]</code>

12.1.82 Class `basic_streambuf<char, char_traits<char> >`

12.1.82.1 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 12-131

Table 12-131 Primary vtable for `basic_streambuf<char, char_traits<char> >`

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>basic_streambuf<char, char_traits<char> ></code>

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]:	basic_streambuf<char, char_traits<char> >::setbuf(char*, long)
vfunc[4]:	basic_streambuf<char, char_traits<char> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	basic_streambuf<char, char_traits<char> >::xsgetn(char*, long)
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]:	basic_streambuf<char, char_traits<char> >::xspn(char const*, long)
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 12-132

Table 12-132 typeid 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> >
------	---

12.1.82.2 Interfaces for Class basic_streambuf<char, char_traits<char> >

An LSB conforming implementation shall provide the architecture specific methods for Class std::basic_streambuf<char, std::char_traits<char> > specified in Table 12-133, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-133 libstdc++ - Class basic_streambuf<char, char_traits<char> > Function Interfaces

basic_streambuf<char, char_traits<char> >::pubseekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sgetn(char*, long)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::sputn(char const*, long)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::setbuf(char*, long)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::xsgetn(char*, long)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::xsputn(char const*, long)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_streambuf<char, char_traits<char> >::pubsetbuf(char*, long)(GLIBCXX_3.4) [ISOCXX]

12.1.83 Class basic_streambuf<wchar_t, char_traits<wchar_t> >

12.1.83.1 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 12-134

Table 12-134 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]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, long)
vfunc[4]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, long)
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]:	basic_streambuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, long)
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 12-135

Table 12-135 typeinfo for `basic_streambuf<wchar_t, char_traits<wchar_t> >`

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for <code>basic_streambuf<wchar_t, char_traits<wchar_t> ></code>

12.1.83.2 Interfaces for Class `basic_streambuf<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_streambuf<wchar_t, std::char_traits<wchar_t> >` specified in Table 12-136, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-136 `libstdc++` - Class `basic_streambuf<wchar_t, char_traits<wchar_t> >` > Function Interfaces

<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::pubseekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::sgetn(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::sputn(wchar_t const*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]</code>
<code>basic_streambuf<wchar_t, char_traits<wchar_t> >::pubsetbuf(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.84 Class `basic_filebuf<char, char_traits<char> >`

12.1.84.1 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 12-137

Table 12-137 Primary vtable for `basic_filebuf<char, char_traits<char> >`

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>basic_filebuf<char, char_traits<char> ></code>

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]:	basic_filebuf<char, char_traits<char>>::~setbuf(char*, long)
vfunc[4]:	basic_filebuf<char, char_traits<char>>::~seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	basic_filebuf<char, char_traits<char>>::~xsgetn(char*, long)
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]:	basic_filebuf<char, char_traits<char>>::~xsputn(char const*, long)
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 12-138

Table 12-138 typeid for `basic_filebuf<char, char_traits<char>>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>basic_filebuf<char, char_traits<char>></code>

12.1.84.2 Interfaces for Class `basic_filebuf<char, char_traits<char>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_filebuf<char, std::char_traits<char>>` specified in

Table 12-139, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-139 libstdc++ - Class basic_filebuf<char, char_traits<char> > Function Interfaces

basic_filebuf<char, char_traits<char> >::_M_set_buffer(long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_convert_to_external(char*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::setbuf(char*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::xsgetn(char*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::xspn(char const*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::_M_seek(long, _Ios_Seekdir, __mbstate_t)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<char, char_traits<char> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]

12.1.85 Class basic_filebuf<wchar_t, char_traits<wchar_t> >

12.1.85.1 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 12-140

Table 12-140 Primary vtable for basic_filebuf<wchar_t, char_traits<wchar_t> >

Base Offset	0
Virtual Base Offset	0
RTTI	typeid 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]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, long)

vfunc[4]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)
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]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, long)
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]:	basic_filebuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, long)
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 12-141

Table 12-141 typeid for `basic_filebuf<wchar_t, char_traits<wchar_t> >`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>basic_filebuf<wchar_t, char_traits<wchar_t> ></code>

12.1.85.2 Interfaces for Class `basic_filebuf<wchar_t, char_traits<wchar_t> >`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::basic_filebuf<wchar_t, std::char_traits<wchar_t> >` specified in Table 12-142, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-142 libstdcxx - Class basic_filebuf<wchar_t, char_traits<wchar_t> > Function Interfaces

basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_set_buffer(long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_convert_to_external(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::setbuf(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::xsgetn(wchar_t*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::xsputn(wchar_t const*, long)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::_M_seek(long, _Ios_Seekdir, __mbstate_t)(GLIBCXX_3.4) [ISOCXX]
basic_filebuf<wchar_t, char_traits<wchar_t> >::seekoff(long, _Ios_Seekdir, _Ios_Openmode)(GLIBCXX_3.4) [ISOCXX]
basic_ostream<wchar_t, char_traits<wchar_t> >::_M_write(wchar_t const*, long)(GLIBCXX_3.4) [ISOCXX]
virtual thunk to basic_fstream<wchar_t, char_traits<wchar_t> >::~basic_fstream()(GLIBCXX_3.4) [CXXABI1.86]

12.1.86 Class ios_base

12.1.86.1 Class data for ios_base

The virtual table for the std::ios_base class is described in the generic part of this specification.

The Run Time Type Information for the std::ios_base class is described by Table 12-143

Table 12-143 typeinfo for ios_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for ios_base

12.1.86.2 Interfaces for Class ios_base

No external methods are defined for libstdcxx - Class std::ios_base in this part of the specification. See also the generic specification.

12.1.87 Class basic_ios<char, char_traits<char> >

12.1.87.1 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 in the generic part of this specification.

12.1.87.2 Interfaces for Class `basic_ios<char, char_traits<char>>`

No external methods are defined for libstdc++ - Class `std::basic_ios<char, std::char_traits<char>>` in this part of the specification. See also the generic specification.

12.1.88 Class `basic_ios<wchar_t, char_traits<wchar_t>>`

12.1.88.1 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 in the generic part of this specification.

The Run Time Type Information for the `std::basic_ios<wchar_t, std::char_traits<wchar_t>>` class is described by Table 12-144

Table 12-144 typeinfo for `basic_ios<wchar_t, char_traits<wchar_t>>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_t</code> type_info	1026
Name	typeinfo name for <code>basic_ios<wchar_t, char_traits<wchar_t>></code>	
flags:	8	
basetype:	typeinfo for <code>ios_base</code>	

12.1.88.2 Interfaces for Class `basic_ios<wchar_t, char_traits<wchar_t>>`

No external methods are defined for libstdc++ - Class `std::basic_ios<wchar_t, std::char_traits<wchar_t>>` in this part of the specification. See also the generic specification.

12.1.89 Class `ios_base::failure`

12.1.89.1 Class data for `ios_base::failure`

The virtual table for the `std::ios_base::failure` class is described in the generic part of this specification.

The Run Time Type Information for the `std::ios_base::failure` class is described by Table 12-145

Table 12-145 typeinfo for `ios_base::failure`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>ios_base::failure</code>

12.1.89.2 Interfaces for Class `ios_base::failure`

No external methods are defined for libstdc++ - Class `std::ios_base::failure` in this part of the specification. See also the generic specification.

12.1.90 Class `__timepunct<char>`

12.1.90.1 Class data for `__timepunct<char>`

The virtual table for the `std::__timepunct<char>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::__timepunct<char>` class is described by Table 12-146

Table 12-146 typeinfo for `__timepunct<char>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>__timepunct<char></code>

12.1.90.2 Interfaces for Class `__timepunct<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::__timepunct<char>` specified in Table 12-147, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-147 libstdc++ - Class `__timepunct<char>` Function Interfaces

<code>__timepunct<char>::M_put(char*, unsigned long, char const*, tm const*)</code> const(GLIBCXX_3.4) [ISOCXX]
<code>__timepunct<char>::__timepunct(__locale_struct*, char const*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>__timepunct<char>::__timepunct(__timepunct_cache<char>*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>__timepunct<char>::__timepunct(unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>__timepunct<char>::__timepunct(__locale_struct*, char const*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>__timepunct<char>::__timepunct(__timepunct_cache<char>*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>__timepunct<char>::__timepunct(unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]

12.1.91 Class `__timepunct<wchar_t>`

12.1.91.1 Class data for `__timepunct<wchar_t>`

The virtual table for the `std::__timepunct<wchar_t>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::__timepunct<wchar_t>` class is described by Table 12-148

Table 12-148 typeinfo for `__timepunct<wchar_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
-------------	---

Name	typeinfo name for __timepunct<wchar_t>
------	---

12.1.91.2 Interfaces for Class __timepunct<wchar_t>

An LSB conforming implementation shall provide the architecture specific methods for Class std::__timepunct<wchar_t> specified in Table 12-149, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-149 libstdcxx - Class __timepunct<wchar_t> Function Interfaces

__timepunct<wchar_t>::M_put(wchar_t*, unsigned long, wchar_t const*, tm const*) const (GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::__timepunct(__locale_struct*, char const*, unsigned long) (GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::__timepunct(__timepunct_cache<wchar_t>*, unsigned long) (GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::__timepunct(unsigned long) (GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::__timepunct(__locale_struct*, char const*, unsigned long) (GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::__timepunct(__timepunct_cache<wchar_t>*, unsigned long) (GLIBCXX_3.4) [ISOCXX]
__timepunct<wchar_t>::__timepunct(unsigned long) (GLIBCXX_3.4) [ISOCXX]

12.1.92 Class messages_base

12.1.92.1 Class data for messages_base

The Run Time Type Information for the std::messages_base class is described by Table 12-150

Table 12-150 typeinfo for messages_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for messages_base

12.1.92.2 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 generic specification.

12.1.93 Class messages<char>

12.1.93.1 Class data for messages<char>

The virtual table for the std::messages<char> class is described in the generic part of this specification.

12.1.93.2 Interfaces for Class messages<char>

An LSB conforming implementation shall provide the architecture specific methods for Class std::messages<char> specified in Table 12-151, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-151 libstdcxx - Class messages<char> Function Interfaces

messages<char>::messages(__locale_struct*, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
messages<char>::messages(unsigned long)(GLIBCXX_3.4) [ISOCXX]
messages<char>::messages(__locale_struct*, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
messages<char>::messages(unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.94 Class messages<wchar_t>

12.1.94.1 Class data for messages<wchar_t>

The virtual table for the std::messages<wchar_t> class is described in the generic part of this specification.

12.1.94.2 Interfaces for Class messages<wchar_t>

An LSB conforming implementation shall provide the architecture specific methods for Class std::messages<wchar_t> specified in Table 12-152, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-152 libstdcxx - Class messages<wchar_t> Function Interfaces

messages<wchar_t>::messages(__locale_struct*, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::messages(unsigned long)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::messages(__locale_struct*, char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
messages<wchar_t>::messages(unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.95 Class messages_byname<char>

12.1.95.1 Class data for messages_byname<char>

The virtual table for the std::messages_byname<char> class is described in the generic part of this specification.

The Run Time Type Information for the std::messages_byname<char> class is described by Table 12-153

Table 12-153 typeid for messages_byname<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for messages_byname<char>

12.1.95.2 Interfaces for Class `messages_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::messages_byname<char>` specified in Table 12-154, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-154 libstdcxx - Class `messages_byname<char>` Function Interfaces

<code>messages_byname<char>::messages_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>messages_byname<char>::messages_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.96 Class `messages_byname<wchar_t>`

12.1.96.1 Class data for `messages_byname<wchar_t>`

The virtual table for the `std::messages_byname<wchar_t>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::messages_byname<wchar_t>` class is described by Table 12-155

Table 12-155 typeinfo for `messages_byname<wchar_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>messages_byname<wchar_t></code>

12.1.96.2 Interfaces for Class `messages_byname<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::messages_byname<wchar_t>` specified in Table 12-156, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-156 libstdcxx - Class `messages_byname<wchar_t>` Function Interfaces

<code>messages_byname<wchar_t>::messages_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>messages_byname<wchar_t>::messages_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.97 Class `numpunct<char>`

12.1.97.1 Class data for `numpunct<char>`

The virtual table for the `std::numpunct<char>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::numpunct<char>` class is described by Table 12-157

Table 12-157 typeinfo for `numpunct<char>`

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for <code>numpunct<char></code>

12.1.97.2 Interfaces for Class `numpunct<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::numpunct<char>` specified in Table 12-158, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-158 `libstdcxx` - Class `numpunct<char>` Function Interfaces

<code>numpunct<char>::numpunct(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<char>::numpunct(__numpunct_cache<char>*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<char>::numpunct(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<char>::numpunct(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<char>::numpunct(__numpunct_cache<char>*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<char>::numpunct(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.98 Class `numpunct<wchar_t>`

12.1.98.1 Class data for `numpunct<wchar_t>`

The virtual table for the `std::numpunct<wchar_t>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::numpunct<wchar_t>` class is described by Table 12-159

Table 12-159 typeinfo for `numpunct<wchar_t>`

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeinfo name for <code>numpunct<wchar_t></code>

12.1.98.2 Interfaces for Class `numpunct<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::numpunct<wchar_t>` specified in Table 12-160, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-160 `libstdcxx` - Class `numpunct<wchar_t>` Function Interfaces

<code>numpunct<wchar_t>::numpunct(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

<code>numpunct<wchar_t>::numpunct(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<wchar_t>::numpunct(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct<wchar_t>::numpunct(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.99 Class `numpunct_byname<char>`

12.1.99.1 Class data for `numpunct_byname<char>`

The virtual table for the `std::numpunct_byname<char>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::numpunct_byname<char>` class is described by Table 12-161

Table 12-161 typeinfo for `numpunct_byname<char>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>numpunct_byname<char></code>

12.1.99.2 Interfaces for Class `numpunct_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::numpunct_byname<char>` specified in Table 12-162, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-162 libstdc++ - Class `numpunct_byname<char>` Function Interfaces

<code>numpunct_byname<char>::numpunct_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct_byname<char>::numpunct_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.100 Class `numpunct_byname<wchar_t>`

12.1.100.1 Class data for `numpunct_byname<wchar_t>`

The virtual table for the `std::numpunct_byname<wchar_t>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::numpunct_byname<wchar_t>` class is described by Table 12-163

Table 12-163 typeinfo for `numpunct_byname<wchar_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>numpunct_byname<wchar_t></code>

12.1.100.2 Interfaces for Class `numpunct_byname<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::numpunct_byname<wchar_t>` specified in Table 12-164, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-164 `libstdcxx` - Class `numpunct_byname<wchar_t>` Function Interfaces

<code>numpunct_byname<wchar_t>::numpunct_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>numpunct_byname<wchar_t>::numpunct_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.101 Class `__codecvt_abstract_base<char, char, __mbstate_t>`**12.1.101.1 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 in the generic part of this specification.

12.1.101.2 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 generic specification.

12.1.102 Class `__codecvt_abstract_base<wchar_t, char, __mbstate_t>`**12.1.102.1 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 in the generic part of this specification.

12.1.102.2 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 generic specification.

12.1.103 Class `codecvt_base`**12.1.103.1 Class data for `codecvt_base`**

The Run Time Type Information for the `std::codecvt_base` class is described by Table 12-165

Table 12-165 typeinfo for codecvt_base

Base Vtable	vtable for __cxxabiv1::__class_type_info
Name	typeinfo name for codecvt_base

12.1.103.2 Interfaces for Class codecvt_base

No external methods are defined for libstdc++ - Class std::codecvt_base in this part of the specification. See also the generic specification.

12.1.104 Class codecvt<char, char, __mbstate_t>

12.1.104.1 Class data for codecvt<char, char, __mbstate_t>

The virtual table for the std::codecvt<char, char, __mbstate_t> class is described by Table 12-166

Table 12-166 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]:	codecvt<char, char, __mbstate_t>::do_length(__mbstate_t&, char const*, char const*, unsigned long) const

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 12-167

Table 12-167 typeinfo for `codecvt<char, char, __mbstate_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>codecvt<char, char, __mbstate_t></code>

12.1.104.2 Interfaces for Class `codecvt<char, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt<char, char, __mbstate_t>` specified in Table 12-168, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-168 libstdcxx - Class `codecvt<char, char, __mbstate_t>` Function Interfaces

<code>codecvt<char, char, __mbstate_t>::do_length(__mbstate_t&, char const*, char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt<char, char, __mbstate_t>::codecvt(__locale_struct*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt<char, char, __mbstate_t>::codecvt(unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt<char, char, __mbstate_t>::codecvt(__locale_struct*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>codecvt<char, char, __mbstate_t>::codecvt(unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]

12.1.105 Class `codecvt<wchar_t, char, __mbstate_t>`

12.1.105.1 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 12-169

Table 12-169 Primary vtable for `codecvt<wchar_t, char, __mbstate_t>`

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>codecvt<wchar_t, char, __mbstate_t></code>
vfunc[0]:	<code>codecvt<wchar_t, char, __mbstate_t>::~~codecvt()</code>

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]:	codecvt<wchar_t, char, __mbstate_t>::~do_length(__mbstate_t&, char const*, char const*, unsigned long) const
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 12-170

Table 12-170 typeinfo for `codecvt<wchar_t, char, __mbstate_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>codecvt<wchar_t, char, __mbstate_t></code>

12.1.105.2 Interfaces for Class `codecvt<wchar_t, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt<wchar_t, char, __mbstate_t>` specified in Table 12-171, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-171 `libstdc++` - Class `codecvt<wchar_t, char, __mbstate_t>` Function Interfaces

<code>codecvt<wchar_t, char, __mbstate_t>::~do_length(__mbstate_t&, char const*, char const*, unsigned long) const</code> (GLIBCXX_3.4) [ISOCXX]
--

<code>codecvt<wchar_t, char, __mbstate_t>::codecvt(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt<wchar_t, char, __mbstate_t>::codecvt(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt<wchar_t, char, __mbstate_t>::codecvt(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt<wchar_t, char, __mbstate_t>::codecvt(unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.106 Class `codecvt_byname<char, char, __mbstate_t>`

12.1.106.1 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 12-172

Table 12-172 Primary vtable for `codecvt_byname<char, char, __mbstate_t>`

Base Offset	0
Virtual Base Offset	0
RTTI	<code>typeinfo for codecvt_byname<char, char, __mbstate_t></code>
<code>vfunc[0]:</code>	<code>codecvt_byname<char, char, __mbstate_t>::~~codecvt_byname()</code>
<code>vfunc[1]:</code>	<code>codecvt_byname<char, char, __mbstate_t>::~~codecvt_byname()</code>
<code>vfunc[2]:</code>	<code>codecvt<char, char, __mbstate_t>::do_out(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const</code>
<code>vfunc[3]:</code>	<code>codecvt<char, char, __mbstate_t>::do_unshift(__mbstate_t&, char*, char*, char*&) const</code>
<code>vfunc[4]:</code>	<code>codecvt<char, char, __mbstate_t>::do_in(__mbstate_t&, char const*, char const*, char const*&, char*, char*, char*&) const</code>
<code>vfunc[5]:</code>	<code>codecvt<char, char, __mbstate_t>::do_encoding() const</code>
<code>vfunc[6]:</code>	<code>codecvt<char, char, __mbstate_t>::do_always_noconv() const</code>
<code>vfunc[7]:</code>	<code>codecvt<char, char, __mbstate_t>::do_length(__mbstate_t</code>

	&, char const*, char const*, unsigned long) const
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 12-173

Table 12-173 typeinfo for `codecvt_byname<char, char, __mbstate_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>codecvt_byname<char, char, __mbstate_t></code>

12.1.106.2 Interfaces for Class `codecvt_byname<char, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt_byname<char, char, __mbstate_t>` specified in Table 12-174, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-174 `libstdc++` - Class `codecvt_byname<char, char, __mbstate_t>` Function Interfaces

<code>codecvt_byname<char, char, __mbstate_t>::codecvt_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>codecvt_byname<char, char, __mbstate_t>::codecvt_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.107 Class `codecvt_byname<wchar_t, char, __mbstate_t>`

12.1.107.1 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 12-175

Table 12-175 Primary vtable for `codecvt_byname<wchar_t, char, __mbstate_t>`

Base Offset	0
Virtual Base Offset	0
RTTI	typeinfo for <code>codecvt_byname<wchar_t, char, __mbstate_t></code>
vfunc[0]:	<code>codecvt_byname<wchar_t, char, __mbstate_t>::~~codecvt_byname()</code>

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]:	codecvt<wchar_t, char, __mbstate_t>::do_length(__mbstate_t&, char const*, char const*, unsigned long) const
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 12-176

Table 12-176 typeid for `codecvt_byname<wchar_t, char, __mbstate_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeid name for <code>codecvt_byname<wchar_t, char, __mbstate_t></code>

12.1.107.2 Interfaces for Class `codecvt_byname<wchar_t, char, __mbstate_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::codecvt_byname<wchar_t, char, __mbstate_t>` specified in Table 12-177, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-177 libstdcxx - Class codecvt_byname<wchar_t, char, __mbstate_t> Function Interfaces

codecvt_byname<wchar_t, char, __mbstate_t>::codecvt_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
codecvt_byname<wchar_t, char, __mbstate_t>::codecvt_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.108 Class collate<char>**12.1.108.1 Class data for collate<char>**

The virtual table for the std::collate<char> class is described in the generic part of this specification.

The Run Time Type Information for the std::collate<char> class is described by Table 12-178

Table 12-178 typeid for collate<char>

Base Vtable	vtable for __cxxabiv1::__si_class_type_info
Name	typeid name for collate<char>

12.1.108.2 Interfaces for Class collate<char>

An LSB conforming implementation shall provide the architecture specific methods for Class std::collate<char> specified in Table 12-179, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-179 libstdcxx - Class collate<char> Function Interfaces

collate<char>::_M_transform(char*, char const*, unsigned long) const(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(unsigned long)(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(__locale_struct*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
collate<char>::collate(unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.109 Class collate<wchar_t>**12.1.109.1 Class data for collate<wchar_t>**

The virtual table for the std::collate<wchar_t> class is described in the generic part of this specification.

The Run Time Type Information for the std::collate<wchar_t> class is described by Table 12-180

Table 12-180 typeinfo for `collate<wchar_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>collate<wchar_t></code>

12.1.109.2 Interfaces for Class `collate<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::collate<wchar_t>` specified in Table 12-181, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-181 `libstdcxx` - Class `collate<wchar_t>` Function Interfaces

<code>collate<wchar_t>::M_transform(wchar_t*, wchar_t const*, unsigned long)</code> <code>const</code> (GLIBCXX_3.4) [ISOCXX]
<code>collate<wchar_t>::collate(__locale_struct*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>collate<wchar_t>::collate(unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>collate<wchar_t>::collate(__locale_struct*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>collate<wchar_t>::collate(unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]

12.1.110 Class `collate_byname<char>`

12.1.110.1 Class data for `collate_byname<char>`

The virtual table for the `std::collate_byname<char>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::collate_byname<char>` class is described by Table 12-182.

Table 12-182 typeinfo for `collate_byname<char>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>collate_byname<char></code>

12.1.110.2 Interfaces for Class `collate_byname<char>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::collate_byname<char>` specified in Table 12-183, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-183 `libstdcxx` - Class `collate_byname<char>` Function Interfaces

<code>collate_byname<char>::collate_byname(char const*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]
<code>collate_byname<char>::collate_byname(char const*, unsigned long)</code> (GLIBCXX_3.4) [ISOCXX]

12.1.111 Class `collate_byname<wchar_t>`

12.1.111.1 Class data for `collate_byname<wchar_t>`

The virtual table for the `std::collate_byname<wchar_t>` class is described in the generic part of this specification.

The Run Time Type Information for the `std::collate_byname<wchar_t>` class is described by Table 12-184

Table 12-184 typeinfo for `collate_byname<wchar_t>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>collate_byname<wchar_t></code>

12.1.111.2 Interfaces for Class `collate_byname<wchar_t>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::collate_byname<wchar_t>` specified in Table 12-185, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-185 `libstdcxx` - Class `collate_byname<wchar_t>` Function Interfaces

<code>collate_byname<wchar_t>::collate_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>collate_byname<wchar_t>::collate_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.112 Class `time_base`

12.1.112.1 Class data for `time_base`

The Run Time Type Information for the `std::time_base` class is described by Table 12-186

Table 12-186 typeinfo for `time_base`

Base Vtable	vtable for <code>__cxxabiv1::__class_type_info</code>
Name	typeinfo name for <code>time_base</code>

12.1.112.2 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 generic specification.

12.1.113 Class `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>`

12.1.113.1 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 in the generic part of this specification.

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 12-187

Table 12-187 `typeinfo` for `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	<code>typeinfo</code> name for <code>time_get_byname<char, istreambuf_iterator<char, char_traits<char>>></code>

12.1.113.2 Interfaces for Class `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_get_byname<char, std::istreambuf_iterator<char, std::char_traits<char>>>` specified in Table 12-188, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-188 `libstdc++` - Class `time_get_byname<char, istreambuf_iterator<char, char_traits<char>>>` Function Interfaces

<code>time_get_byname<char, istreambuf_iterator<char, char_traits<char>>></code> <code>>::time_get_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
<code>time_get_byname<char, istreambuf_iterator<char, char_traits<char>>></code> <code>>::time_get_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>

12.1.114 Class `time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

12.1.114.1 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 in the generic part of this specification.

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 12-189

Table 12-189 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>>>

12.1.114.2 Interfaces for Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>

An LSB conforming implementation shall provide the architecture specific methods for Class std::time_get_byname<wchar_t, std::istreambuf_iterator<wchar_t, std::char_traits<wchar_t>>> specified in Table 12-190, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-190 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(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]
time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t>>>::time_get_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]

12.1.115 Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>>

12.1.115.1 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 in the generic part of this specification.

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 12-191

Table 12-191 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>>>

12.1.115.2 Interfaces for Class `time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_put_byname<char, std::ostreambuf_iterator<char, std::char_traits<char>>>` specified in Table 12-192, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-192 `libstdcxx` - Class `time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>>` Function Interfaces

<code>time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>></code> <code>>::time_put_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
--

<code>time_put_byname<char, ostreambuf_iterator<char, char_traits<char>>></code> <code>>::time_put_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
--

12.1.116 Class `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

12.1.116.1 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 in the generic part of this specification.

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 12-193

Table 12-193 `typeinfo` for `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

Base Vtable	vtable for <code>__cxxabiv1::__si_class_type_info</code>
Name	typeinfo name for <code>time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>></code>

12.1.116.2 Interfaces for Class `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>`

An LSB conforming implementation shall provide the architecture specific methods for Class `std::time_put_byname<wchar_t, std::ostreambuf_iterator<wchar_t, std::char_traits<wchar_t>>>` specified in Table 12-194, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-194 `libstdcxx` - Class `time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>>` Function Interfaces

<code>time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t>>></code> <code>>::time_put_byname(char const*, unsigned long)(GLIBCXX_3.4) [ISOCXX]</code>
