INTERNATIONAL STANDARD

ISO/IEC 23360-2

First edition
2006-12-15

Linux Standard Base (LSB) core specification 3.1 —

Part 2: Specification for IA32 architecture

Spécifications 3.1 relatives au noyau de base normalisé Linux (LSB) —

Partie 2: Spécifications pour l'architecture IA32

Reference number
ISO/IEC 23360-2:2006(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Linux Standard Base Core Specification for IA32 3.1
ISO/IEC 23360-2:2006(E)

Contents

List of Tables

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

International Standard ISO/IEC 23360-2 was prepared by the Free Standards Group and was adopted, under the PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

ISO/IEC 23360 consists of the following parts, under the general title *Linux Standard Base (LSB) core specification 3.1*:

— *Part 1: Generic specification*

— *Part 2: Specification for IA32 architecture*

— *Part 3: Specification for IA64 architecture*

— *Part 4: Specification for AMD64 architecture*

— *Part 5: Specification for PPC32 architecture*

— *Part 6: Specification for PPC64 architecture*

— *Part 7: Specification for S390 architecture*

— *Part 8: Specification for S390X architecture*

Introduction

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

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

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

- The first number (x) is the major version number. All versions with the same major version number should share binary compatibility. Any addition or deletion of a new library results in a new version number. Interfaces marked as `deprecated` may be removed from the specification at a major version change.
- The second number (y) is the minor version number. Individual interfaces may be added if all certified implementations already had that (previously undocumented) interface. Interfaces may be marked as `deprecated` at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.
- The third number (z), if present, is the editorial level. Only editorial changes should be included in such versions.

Since this specification is a descriptive Application Binary Interface, and not a source level API specification, it is not possible to make a guarantee of 100% backward compatibility between major releases. However, it is the intent that those parts of the binary interface that are visible in the source level API will remain backward compatible from version to version, except where a feature marked as `deprecated` in one release may be removed from a future release.

Implementors are strongly encouraged to make use of symbol versioning to permit simultaneous support of applications conforming to different releases of this specification.

This is version 3.1 of the Linux Standard Base Core Specification. This specification is part of a family of specifications under the general title "Linux Standard Base (LSB) core specification 3.1". Developers of applications or implementations interested in using the LSB trademark should see the Free Standards Group Certification Policy for details.

I Introductory Elements

Linux Standard Base (LSB) core specification 3.1 —

Part 2:
Specification for IA32 architecture

1 Scope

1.1 General

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

These specifications are composed of two basic parts: A common specification ("LSB-generic" or "generic LSB"), ISO/IEC 23360-1, describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific part ("LSB-arch" or "archLSB") describing the parts of the interface that vary by processor architecture. Together, the LSB-generic and the relevant architecture-specific part of ISO/IEC 23360 for a single hardware architecture provide a complete interface specification for compiled application programs on systems that share a common hardware architecture.

ISO/IEC 23360-1, the LSB-generic document, should be used in conjunction with an architecture-specific part. Whenever a section of the LSB-generic specification is supplemented by architecture-specific information, the LSB-generic document includes a reference to the architecture part. Architecture-specific parts of ISO/IEC 23360 may also contain additional information that is not referenced in the LSB-generic document.

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

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

1.2 Module Specific Scope

This is the IA32 architecture specific Core part of the Linux Standard Base (LSB). This part supplements the generic LSB Core module with those interfaces that differ between architectures.

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

2 References

2.1 Normative References

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

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

Table 2-1 Normative References

Name	Title	URL
ISO/IEC 23360-1	ISO/IEC 23360-1:2006, *Linux Standard Base (LSB) core specification 3.1 — Part 1: Generic Specification*	http://www.linuxbase.org/spec/
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
Intel® Architecture Software Developer's Manual Volume 1	The IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture	http://developer.intel.com/design/pentium4/manuals/245470.htm
Intel® Architecture Software Developer's Manual Volume 2	The IA-32 Intel® Architecture Software Developer's Manual Volume 2: Instruction Set Reference	http://developer.intel.com/design/pentium4/manuals/245471.htm
Intel® Architecture Software Developer's Manual Volume 3	The IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide	http://developer.intel.com/design/pentium4/manuals/245472.htm
ISO C (1999)	ISO/IEC 9899: 1999, *Programming Languages — C*	
ISO POSIX (2003)	ISO/IEC 9945-1:2003, *Information technology — Portable Operating System Interface (POSIX) —Part 1: Base Definitions*	http://www.unix.org/version3/

Name	Title	URL
	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*	
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
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	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
System V ABI, IA32 Supplement	System V Application Binary Interface - Intel386 Architecture Processor Supplement, Fourth Edition	http://www.caldera.com/developers/devspecs/abi386-4.pdf

Name	Title	URL
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	http://www.opengroup.org/publications/catalog/un.htm

2.2 Informative References/Bibliography

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

Table 2-2 Other References

Name	Title	URL
DWARF Debugging Information Format, Revision 2.0.0	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf
DWARF Debugging Information Format, Revision 3.0.0 (Draft)	DWARF Debugging Information Format, Revision 3.0.0 (Draft)	http://refspecs.freestandards.org/dwarf/
IEC 60559/IEEE 754 Floating Point	IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems*	http://www.ieee.org/
ISO/IEC TR 14652	ISO/IEC TR 14652:2004, *Information technology — 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	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt

Name	Title	URL
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 1831/1832 RPC & XDR	IETF RFC 1831 & 1832	http://www.ietf.org/
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1950: ZLIB Compressed Data Format Specication	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821: Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822: Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 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
SUSv2 Commands and Utilities	The Single UNIX Specification (SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/publications/catalog/un.htm
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on IA32 Linux Standard Base systems, with the specified runtime names. These names override or supplement the names specified in the generic LSB (ISO/IEC 23360-1) 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
libm	libm.so.6
libdl	libdl.so.2
libcrypt	libcrypt.so.1
libz	libz.so.1
libncurses	libncurses.so.5
libutil	libutil.so.1
libc	libc.so.6
libpthread	libpthread.so.0
proginterp	/lib/ld-lsb.so.3
libgcc_s	libgcc_s.so.1

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

3.2 LSB Implementation Conformance

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

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

A conforming implementation shall satisfy the following requirements:

- A processor architecture represents a family of related processors which may not have identical feature sets. The architecture specific parts of ISO/IEC 23360 that supplement this specification for a given target processor architecture describe a minimum acceptable processor. The implementation shall provide all features of this processor, whether in hardware or through emulation transparent to the application.
- The implementation shall be capable of executing compiled applications having the format and using the system interfaces described in this document.
- The implementation shall provide libraries containing the interfaces specified by this document, and shall provide a dynamic linking mechanism that allows

these interfaces to be attached to applications at runtime. All the interfaces shall behave as specified in this document.

- The map of virtual memory provided by the implementation shall conform to the requirements of this document.
- The implementation's low-level behavior with respect to function call linkage, system traps, signals, and other such activities shall conform to the formats described in this document.
- The implementation shall provide all of the mandatory interfaces in their entirety.
- The implementation may provide one or more of the optional interfaces. Each optional interface that is provided shall be provided in its entirety. The product documentation shall state which optional interfaces are provided.
- The implementation shall provide all files and utilities specified as part of this document in the format defined here and in other referenced documents. All commands and utilities shall behave as required by this document. The implementation shall also provide all mandatory components of an application's runtime environment that are included or referenced in this document.
- The implementation, when provided with standard data formats and values at a named interface, shall provide the behavior defined for those values and data formats at that interface. However, a conforming implementation may consist of components which are separately packaged and/or sold. For example, a vendor of a conforming implementation might sell the hardware, operating system, and windowing system as separately packaged items.
- The implementation may provide additional interfaces with different names. It may also provide additional behavior corresponding to data values outside the standard ranges, for standard named interfaces.

3.3 LSB Application Conformance

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

A conforming application shall satisfy the following requirements:

- Its executable files shall be either shell scripts or object files in the format defined for the Object File Format system interface.
- Its object files shall participate in dynamic linking as defined in the Program Loading and Linking System interface.
- It shall employ only the instructions, traps, and other low-level facilities defined in the Low-Level System interface as being for use by applications.
- If it requires any optional interface defined in this document in order to be installed or to execute successfully, the requirement for that optional interface shall be stated in the application's documentation.
- It shall not use any interface or data format that is not required to be provided by a conforming implementation, unless:
  - If such an interface or data format is supplied by another application through direct invocation of that application during execution, that application shall be in turn an LSB conforming application.

- The use of that interface or data format, as well as its source, shall be identified in the documentation of the application.
- It shall not use any values for a named interface that are reserved for vendor extensions.

A strictly conforming application shall not require or use any interface, facility, or implementation-defined extension that is not defined in this document in order to be installed or to execute successfully.

4 Definitions

For the purposes of this document, the following definitions, as specified in the *ISO/IEC Directives, Part 2, 2004, 5th Edition*, apply:

can

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

cannot

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

may

is permitted; is allowed; is permissible

need not

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

shall

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

shall not

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

should

it is recommended that; ought to

should not

it is not recommended that; ought not to

5 Terminology

For the purposes of this document, the following terms apply:

archLSB

The architectural part of the LSB Specification which describes the specific parts of the interface that are platform specific. The archLSB is complementary to the gLSB.

Binary Standard

The total set of interfaces that are available to be used in the compiled binary code of a conforming application.

gLSB

The common part of the LSB Specification that describes those parts of the interface that remain constant across all hardware implementations of the LSB.

implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

Source Standard

The set of interfaces that are available to be used in the source code of a conforming application.

undefined

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

Other terms and definitions used in this document shall have the same meaning as defined in Chapter 3 of the Base Definitions volume of ISO POSIX (2003).

6 Documentation Conventions

Throughout this document, the following typographic conventions are used:

`function()`

the name of a function

command

the name of a command or utility

`CONSTANT`

a constant value

`parameter`

a parameter

`variable`

a variable

Throughout this specification, several tables of interfaces are presented. Each entry in these tables has the following format:

name

the name of the interface

(symver)

An optional symbol version identifier, if required.

[*`refno`*]

A reference number indexing the table of referenced specifications that follows this table.

For example,

forkpty(GLIBC_2.0) [SUSv3]

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

Note: Symbol versions are defined in the architecture specific parts of ISO/IEC 23360 only.

II Executable and Linking Format (ELF)

7 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 System V ABI, IA32 Supplement, and is intended to document additions made since the publication of that document.

8 Low Level System Information

8.1 Machine Interface

8.1.1 Processor Architecture

The IA32 Architecture is specified by the following documents

- Intel® Architecture Software Developer's Manual Volume 1
- Intel® Architecture Software Developer's Manual Volume 2
- Intel® Architecture Software Developer's Manual Volume 3

Only the features of the Intel486 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 a conforming application shall 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.

Applications conforming to this specification shall 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 instruction set 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.

8.1.2 Data Representation

LSB-conforming applications shall use the data representation as defined in Chapter 3 of the System V ABI, IA32 Supplement.

8.1.2.1 Byte Ordering

LSB-conforming systems and applications shall use the bit and byte ordering rules specified in Section 1.3.1 of the Intel® Architecture Software Developer's Manual Volume 1.

8.1.2.2 Fundamental Types

In addition to the fundamental types specified in Chapter 3 of the System V ABI, IA32 Supplement, a 64 bit data type is defined here.

Table 8-1 Scalar Types

Type	C	`sizeof`	Alignment (bytes)	Intel386 Architecture
Integral	long long	8	4	signed dou-

Type	C	sizeof	Alignment (bytes)	Intel386 Architecture
				ble word
	signed long long			
	unsigned long long	8	4	unsigned double word

8.1.2.3 Aggregates and Unions

LSB-conforming implementations shall support aggregates and unions with alignment and padding as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.1.2.4 Bit Fields

LSB-conforming implementations shall support structure and union definitions that include bit-fields as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2 Function Calling Sequence

LSB-conforming applications shall use the function calling sequence as defined in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.1 Registers

LSB-conforming applications shall use the general registers provided by the architecture in the manner described in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.2 Floating Point Registers

LSB-conforming applications shall use the floating point registers provided by the architecture in the manner described in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.3 Stack Frame

LSB-conforming applications shall use the stack frame in the manner specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4 Arguments

8.2.4.1 Integral/Pointer

Integral and pointer arguments to functions shall be passed as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4.2 Floating Point

Floating point arguments to functions shall be passed as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4.3 Struct and Union Arguments

Structure and union arguments to functions shall be passed as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4.4 Variable Arguments

As described in Chapter 3 of the System V ABI, IA32 Supplement, LSB-conforming applications using variable argument lists shall use the facilities defined in the header file `<stdarg.h>` to deal with variable argument lists.

> **Note:** This is a requirement of ISO C (1999) and ISO POSIX (2003) as well as System V ABI, IA32 Supplement.

8.2.5 Return Values

8.2.5.1 Void

As described in chapter 3 of System V ABI, IA32 Supplement, functions returning no value need not set any register to any particular value.

8.2.5.2 Integral/Pointer

Functions return scalar values (integer or pointer), shall do so as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.5.3 Floating Point

Functions return floating point values shall do so as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.5.4 Struct and Union

Functions that return a structure or union shall do so as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.3 Operating System Interface

LSB-conforming applications shall use the following aspects of the Operating System Interfaces as defined in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1 Virtual Address Space

LSB-conforming implementations shall support the virtual address space described in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1.1 Page Size

LSB-conforming applications should call `sysconf()` to determine the current page size. See also Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1.2 Virtual Address Assignments

LSB-conforming systems shall provide the virtual address space configuration as described in Chapter 3 of the System V ABI, IA32 Supplement (Virtual Address Assignments).

8.3.1.3 Managing the Process Stack

LSB-conforming systems shall manage the process stack as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1.4 Coding Guidlines

LSB-conforming applications should follow the coding guidleines provided in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.2 Processor Execution Mode

LSB-conforming applications shall run in the user-mode ring as described in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.3 Exception Interface

8.3.3.1 Introduction

LSB-conforming system shall provide the exception interface described in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.3.2 Hardware Exception Types

LSB-conforming systems shall map hardware exceptions to signals as described in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.3.3 Software Trap Types

Software generated traps are subject to the limitations described in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.4 Signal Delivery

There are no architecture specific requirements for signal delivery.

8.3.4.1 Signal Handler Interface

There are no architecture specific requirements for the signal handler interface.

8.4 Process Initialization

An LSB-conforming implementation shall cause an application to be initialized as described in the Process Initialization section of Chapter 3 of the System V ABI, IA32 Supplement, and as described below.

8.4.1 Special Registers

The special registers shall be initialized as described in Chapter 3 of the System V ABI, IA32 Supplement.

8.4.2 Process Stack (on entry)

The process stack shall be initialized as described in Chapter 3 of the System V ABI, IA32 Supplement.

8.4.3 Auxilliary Vector

The auxilliary vector shall be initialized as described in Chapter 3 of the System V ABI, IA32 Supplement.

8.4.4 Environment

There are no architecture specific requirements for environment initialization.

8.5 Coding Examples

8.5.1 Introduction

LSB-conforming applications may follow the coding examples provdied in chapter 3 of the System V ABI, IA32 Supplement in order to implement certain fundamental operations.

8.5.2 Code Model Overview/Architecture Constraints

Chapter 3 of the System V ABI, IA32 Supplement provides an overview of the code model.

8.5.3 Position-Independent Function Prologue

LSB-conforming applications using position independent functions may use the techniques described in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.4 Data Objects

LSB-conforming applications accessing non-stack resident data objects may do so as described in Chapter 3 of the System V ABI, IA32 Supplement, including both absolute and position independent data access techniques.

8.5.5 Function Calls

8.5.5.1 Absolute Direct Function Call

LSB-conforming applications using direct function calls with absolute addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.5.2 Absolute Indirect Function Call

LSB-conforming applications using indirect function calls with absolute addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.5.3 Position-Independent Direct Function Call

LSB-conforming applications using direct function calls with position independent addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.5.4 Position-Independent Indirect Function Call

LSB-conforming applications using indirect function calls with position independent addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.6 Branching

LSB-conforming applications may follow the branching examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.6 C Stack Frame

8.6.1 Variable Argument List

As described in Chapter 3 of the System V ABI, IA32 Supplement, LSB-conforming applications using variable argument lists shall use the facilities defined in the header file `<stdarg.h>` to deal with variable argument lists.

> **Note:** This is a requirement of ISO C (1999) and ISO POSIX (2003) as well as System V ABI, IA32 Supplement.

8.6.2 Dynamic Allocation of Stack Space

LSB-conforming applications may allocate space using the stack following the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.7 Debug Information

There are no architecture specific requirements for debugging information for this architecture. LSB-conforming applications may utilize DWARF sections as described in the generic specification.

9 Object Format

9.1 Introduction

LSB-conforming implementations shall support an object file , called Executable and Linking Format (ELF) as defined by the System V ABI , System V ABI Update , System V ABI, IA32 Supplement and as supplemented by the ISO/IEC 23360-1 and the generic LSB specification.

9.2 ELF Header

9.2.1 Machine Information

LSB-conforming applications shall use the Machine Information as defined in Chapter 4 of the System V ABI, IA32 Supplement, including the *e_ident* array members for EI_CLASS and EI_DATA, the processor identification in *e_machine* and flags in *e_flags*. The operating system identification field, in *e_ident[EI_OSABI]* shall be ELFOSABI_NONE (0).

9.3 Special Sections

9.3.1 Special Sections

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

9.3.1.1 ELF Special Sections

The following sections are defined in Chapter 4 of the System V ABI, IA32 Supplement.

Table 9-1 ELF Special Sections

Name	Type	Attributes
.got	SHT_PROGBITS	SHF_ALLOC+SHF_WRITE
.plt	SHT_PROGBITS	SHF_ALLOC+SHF_EXECINSTR

.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.

.plt

This section holds the procedure linkage table.

9.3.1.2 Addition Special Sections

The following additional sections are defined here.

Table 9-2 Additional Special Sections

Name	Type	Attributes
.rel.dyn	SHT_REL	SHF_ALLOC

.rel.dyn

This section holds relocation information, as described in `Relocation'. These relocations are applied to the .dyn section.

9.4 Symbol Table

LSB-conforming applications shall use the Symbol Table section as defined in Chapter 4 of the System V ABI, IA32 Supplement.

9.5 Relocation

9.5.1 Introduction

LSB-conforming implementations shall support Relocation as defined in Chapter 4 of the System V ABI, IA32 Supplement and as described below.

9.5.2 Relocation Types

The relocation types described in Chapter 4 of the System V ABI, IA32 Supplement shall be supported.

10 Program Loading and Dynamic Linking

10.1 Introduction

LSB-conforming implementations shall support the object file information and system actions that create running programs as specified in the System V ABI , System V ABI Update , System V ABI, IA32 Supplement and as supplemented by ISO/IEC 23360-1 and the generic LSB specification.

10.2 Program Header

10.2.1 Introduction

As described in System V ABI Update, the program header is an array of structures, each describing a segment or other information the system needs to prepare the program for execution.

10.2.2 Types

The IA32 architecture does not define any additional program header types beyond those required in the generic LSB Core specification.

10.2.3 Flags

The IA32 architecture does not define any additional program header flags beyond those required in the generic LSB Core specification.

10.3 Program Loading

LSB-conforming systems shall support program loading as defined in Chapter 5 of the System V ABI, IA32 Supplement.

10.4 Dynamic Linking

LSB-conforming systems shall support dynamic linking as defined in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.1 Dynamic Section

The following dynamic entries are defined in the System V ABI, IA32 Supplement.

DT_PLTGOT

On the Intel386 architecture, this entrys d_ptr member gives the address of the first entry in the global offset table.

10.4.2 Global Offset Table

LSB-conforming implementations shall support use of the global offset table as described in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.3 Shared Object Dependencies

There are no architecture specific requirements for shared object dependencies; see the generic LSB-Core specification.

10.4.4 Function Addresses

Function addresses shall behave as specified in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.5 Procedure Linkage Table

LSB-conforming implementations shall support a Procedure Linkage Table as described in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.6 Initialization and Termination Functions

There are no architecture specific requirements for initialization and termination functions; see the generic LSB-Core specification.

III Base Libraries

11 Libraries

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

Interfaces that are unique to the IA32 platform are defined here. This section should be used in conjunction with the corresponding section in the Linux Standard Base Specification.

11.1 Program Interpreter/Dynamic Linker

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

11.2 Interfaces for libc

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

Table 11-1 libc Definition

Library:	libc
SONAME:	libc.so.6

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

[LFS] Large File Support
[LSB] ISO/IEC 23360-1
[SUSv2] SUSv2
[SUSv3] ISO POSIX (2003)
[SVID.3] SVID Issue 3
[SVID.4] SVID Issue 4

11.2.1 RPC

11.2.1.1 Interfaces for RPC

An LSB conforming implementation shall provide the architecture specific functions for RPC specified in Table 11-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-2 libc - RPC Function Interfaces

authnone_create(GLIBC_2.0) [SVID.4]	clnt_create(GLIBC_2.0) [SVID.4]	clnt_pcreateerror(GLIBC_2.0) [SVID.4]	clnt_perrno(GLIBC_2.0) [SVID.4]
clnt_perror(GLIBC_2.0) [SVID.4]	clnt_spcreateerror(GLIBC_2.0) [SVID.4]	clnt_sperrno(GLIBC_2.0) [SVID.4]	clnt_sperror(GLIBC_2.0) [SVID.4]
key_decryptsession(GLIBC_2.1) [SVID.3]	pmap_getport(GLIBC_2.0) [LSB]	pmap_set(GLIBC_2.0) [LSB]	pmap_unset(GLIBC_2.0) [LSB]
svc_getreqset(GLIBC_2.0) [SVID.3]	svc_register(GLIBC_2.0) [LSB]	svc_run(GLIBC_2.0) [LSB]	svc_sendreply(GLIBC_2.0) [LSB]

svcerr_auth(GLIBC_2.0) [SVID.3]	svcerr_decode(GLIBC_2.0) [SVID.3]	svcerr_noproc(GLIBC_2.0) [SVID.3]	svcerr_noprog(GLIBC_2.0) [SVID.3]
svcerr_progvers(GLIBC_2.0) [SVID.3]	svcerr_systemerr(GLIBC_2.0) [SVID.3]	svcerr_weakauth(GLIBC_2.0) [SVID.3]	svctcp_create(GLIBC_2.0) [LSB]
svcudp_create(GLIBC_2.0) [LSB]	xdr_accepted_reply(GLIBC_2.0) [SVID.3]	xdr_array(GLIBC_2.0) [SVID.3]	xdr_bool(GLIBC_2.0) [SVID.3]
xdr_bytes(GLIBC_2.0) [SVID.3]	xdr_callhdr(GLIBC_2.0) [SVID.3]	xdr_callmsg(GLIBC_2.0) [SVID.3]	xdr_char(GLIBC_2.0) [SVID.3]
xdr_double(GLIBC_2.0) [SVID.3]	xdr_enum(GLIBC_2.0) [SVID.3]	xdr_float(GLIBC_2.0) [SVID.3]	xdr_free(GLIBC_2.0) [SVID.3]
xdr_int(GLIBC_2.0) [SVID.3]	xdr_long(GLIBC_2.0) [SVID.3]	xdr_opaque(GLIBC_2.0) [SVID.3]	xdr_opaque_auth(GLIBC_2.0) [SVID.3]
xdr_pointer(GLIBC_2.0) [SVID.3]	xdr_reference(GLIBC_2.0) [SVID.3]	xdr_rejected_reply(GLIBC_2.0) [SVID.3]	xdr_replymsg(GLIBC_2.0) [SVID.3]
xdr_short(GLIBC_2.0) [SVID.3]	xdr_string(GLIBC_2.0) [SVID.3]	xdr_u_char(GLIBC_2.0) [SVID.3]	xdr_u_int(GLIBC_2.0) [LSB]
xdr_u_long(GLIBC_2.0) [SVID.3]	xdr_u_short(GLIBC_2.0) [SVID.3]	xdr_union(GLIBC_2.0) [SVID.3]	xdr_vector(GLIBC_2.0) [SVID.3]
xdr_void(GLIBC_2.0) [SVID.3]	xdr_wrapstring(GLIBC_2.0) [SVID.3]	xdrmem_create(GLIBC_2.0) [SVID.3]	xdrrec_create(GLIBC_2.0) [SVID.3]
xdrrec_eof(GLIBC_2.0) [SVID.3]			

11.2.2 System Calls

11.2.2.1 Interfaces for System Calls

An LSB conforming implementation shall provide the architecture specific functions for System Calls specified in Table 11-3, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-3 libc - System Calls Function Interfaces

__fxstat(GLIBC_2.0) [LSB]	__getpgid(GLIBC_2.0) [LSB]	__lxstat(GLIBC_2.0) [LSB]	__xmknod(GLIBC_2.0) [LSB]
__xstat(GLIBC_2.0) [LSB]	access(GLIBC_2.0) [SUSv3]	acct(GLIBC_2.0) [LSB]	alarm(GLIBC_2.0) [SUSv3]
brk(GLIBC_2.0) [SUSv2]	chdir(GLIBC_2.0) [SUSv3]	chmod(GLIBC_2.0) [SUSv3]	chown(GLIBC_2.1) [SUSv3]
chroot(GLIBC_2.0) [SUSv2]	clock(GLIBC_2.0) [SUSv3]	close(GLIBC_2.0) [SUSv3]	closedir(GLIBC_2.0) [SUSv3]

creat(GLIBC_2.0) [SUSv3]	dup(GLIBC_2.0) [SUSv3]	dup2(GLIBC_2.0) [SUSv3]	execl(GLIBC_2.0) [SUSv3]
execle(GLIBC_2.0) [SUSv3]	execlp(GLIBC_2.0) [SUSv3]	execv(GLIBC_2.0) [SUSv3]	execve(GLIBC_2.0) [SUSv3]
execvp(GLIBC_2.0) [SUSv3]	exit(GLIBC_2.0) [SUSv3]	fchdir(GLIBC_2.0) [SUSv3]	fchmod(GLIBC_2.0) [SUSv3]
fchown(GLIBC_2.0) [SUSv3]	fcntl(GLIBC_2.0) [LSB]	fdatasync(GLIBC_2.0) [SUSv3]	flock(GLIBC_2.0) [LSB]
fork(GLIBC_2.0) [SUSv3]	fstatvfs(GLIBC_2.1) [SUSv3]	fsync(GLIBC_2.0) [SUSv3]	ftime(GLIBC_2.0) [SUSv3]
ftruncate(GLIBC_2.0) [SUSv3]	getcontext(GLIBC_2.1) [SUSv3]	getegid(GLIBC_2.0) [SUSv3]	geteuid(GLIBC_2.0) [SUSv3]
getgid(GLIBC_2.0) [SUSv3]	getgroups(GLIBC_2.0) [SUSv3]	getitimer(GLIBC_2.0) [SUSv3]	getloadavg(GLIBC_2.2) [LSB]
getpagesize(GLIBC_2.0) [SUSv2]	getpgid(GLIBC_2.0) [SUSv3]	getpgrp(GLIBC_2.0) [SUSv3]	getpid(GLIBC_2.0) [SUSv3]
getppid(GLIBC_2.0) [SUSv3]	getpriority(GLIBC_2.0) [SUSv3]	getrlimit(GLIBC_2.2) [SUSv3]	getrusage(GLIBC_2.0) [SUSv3]
getsid(GLIBC_2.0) [SUSv3]	getuid(GLIBC_2.0) [SUSv3]	getwd(GLIBC_2.0) [SUSv3]	initgroups(GLIBC_2.0) [LSB]
ioctl(GLIBC_2.0) [LSB]	kill(GLIBC_2.0) [LSB]	killpg(GLIBC_2.0) [SUSv3]	lchown(GLIBC_2.0) [SUSv3]
link(GLIBC_2.0) [LSB]	lockf(GLIBC_2.0) [SUSv3]	lseek(GLIBC_2.0) [SUSv3]	mkdir(GLIBC_2.0) [SUSv3]
mkfifo(GLIBC_2.0) [SUSv3]	mlock(GLIBC_2.0) [SUSv3]	mlockall(GLIBC_2.0) [SUSv3]	mmap(GLIBC_2.0) [SUSv3]
mprotect(GLIBC_2.0) [SUSv3]	msync(GLIBC_2.0) [SUSv3]	munlock(GLIBC_2.0) [SUSv3]	munlockall(GLIBC_2.0) [SUSv3]
munmap(GLIBC_2.0) [SUSv3]	nanosleep(GLIBC_2.0) [SUSv3]	nice(GLIBC_2.0) [SUSv3]	open(GLIBC_2.0) [SUSv3]
opendir(GLIBC_2.0) [SUSv3]	pathconf(GLIBC_2.0) [SUSv3]	pause(GLIBC_2.0) [SUSv3]	pipe(GLIBC_2.0) [SUSv3]
poll(GLIBC_2.0) [SUSv3]	read(GLIBC_2.0) [SUSv3]	readdir(GLIBC_2.0) [SUSv3]	readdir_r(GLIBC_2.0) [SUSv3]
readlink(GLIBC_2.0) [SUSv3]	readv(GLIBC_2.0) [SUSv3]	rename(GLIBC_2.0) [SUSv3]	rmdir(GLIBC_2.0) [SUSv3]
sbrk(GLIBC_2.0) [SUSv2]	sched_get_priority_max(GLIBC_2.0) [SUSv3]	sched_get_priority_min(GLIBC_2.0) [SUSv3]	sched_getparam(GLIBC_2.0) [SUSv3]
sched_getscheduler(GLIBC_2.0) [SUSv3]	sched_rr_get_interval(GLIBC_2.0) [SUSv3]	sched_setparam(GLIBC_2.0) [SUSv3]	sched_setscheduler(GLIBC_2.0) [SUSv3]
sched_yield(GLI	select(GLIBC_2.0	setcontext(GLIB	setegid(GLIBC_2.

BC_2.0) [SUSv3]	) [SUSv3]	C_2.0) [SUSv3]	0) [SUSv3]
seteuid(GLIBC_2.0) [SUSv3]	setgid(GLIBC_2.0) [SUSv3]	setitimer(GLIBC_2.0) [SUSv3]	setpgid(GLIBC_2.0) [SUSv3]
setpgrp(GLIBC_2.0) [SUSv3]	setpriority(GLIBC_2.0) [SUSv3]	setregid(GLIBC_2.0) [SUSv3]	setreuid(GLIBC_2.0) [SUSv3]
setrlimit(GLIBC_2.2) [SUSv3]	setrlimit64(GLIBC_2.1) [LFS]	setsid(GLIBC_2.0) [SUSv3]	setuid(GLIBC_2.0) [SUSv3]
sleep(GLIBC_2.0) [SUSv3]	statvfs(GLIBC_2.1) [SUSv3]	stime(GLIBC_2.0) [LSB]	symlink(GLIBC_2.0) [SUSv3]
sync(GLIBC_2.0) [SUSv3]	sysconf(GLIBC_2.0) [SUSv3]	time(GLIBC_2.0) [SUSv3]	times(GLIBC_2.0) [SUSv3]
truncate(GLIBC_2.0) [SUSv3]	ulimit(GLIBC_2.0) [SUSv3]	umask(GLIBC_2.0) [SUSv3]	uname(GLIBC_2.0) [SUSv3]
unlink(GLIBC_2.0) [LSB]	utime(GLIBC_2.0) [SUSv3]	utimes(GLIBC_2.0) [SUSv3]	vfork(GLIBC_2.0) [SUSv3]
wait(GLIBC_2.0) [SUSv3]	wait4(GLIBC_2.0) [LSB]	waitpid(GLIBC_2.0) [LSB]	write(GLIBC_2.0) [SUSv3]
writev(GLIBC_2.0) [SUSv3]			

11.2.3 Standard I/O

11.2.3.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the architecture specific functions for Standard I/O specified in Table 11-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-4 libc - Standard I/O Function Interfaces

_IO_feof(GLIBC_2.0) [LSB]	_IO_getc(GLIBC_2.0) [LSB]	_IO_putc(GLIBC_2.0) [LSB]	_IO_puts(GLIBC_2.0) [LSB]
asprintf(GLIBC_2.0) [LSB]	clearerr(GLIBC_2.0) [SUSv3]	ctermid(GLIBC_2.0) [SUSv3]	fclose(GLIBC_2.1) [SUSv3]
fdopen(GLIBC_2.1) [SUSv3]	feof(GLIBC_2.0) [SUSv3]	ferror(GLIBC_2.0) [SUSv3]	fflush(GLIBC_2.0) [SUSv3]
fflush_unlocked(GLIBC_2.0) [LSB]	fgetc(GLIBC_2.0) [SUSv3]	fgetpos(GLIBC_2.2) [SUSv3]	fgets(GLIBC_2.0) [SUSv3]
fgetwc_unlocked(GLIBC_2.2) [LSB]	fileno(GLIBC_2.0) [SUSv3]	flockfile(GLIBC_2.0) [SUSv3]	fopen(GLIBC_2.1) [SUSv3]
fprintf(GLIBC_2.0) [SUSv3]	fputc(GLIBC_2.0) [SUSv3]	fputs(GLIBC_2.0) [SUSv3]	fread(GLIBC_2.0) [SUSv3]
freopen(GLIBC_2.0) [SUSv3]	fscanf(GLIBC_2.0) [LSB]	fseek(GLIBC_2.0) [SUSv3]	fseeko(GLIBC_2.1) [SUSv3]
fsetpos(GLIBC_2.	ftell(GLIBC_2.0)	ftello(GLIBC_2.1)	fwrite(GLIBC_2.0

2) [SUSv3]	[SUSv3]	[SUSv3]	) [SUSv3]
getc(GLIBC_2.0) [SUSv3]	getc_unlocked(GLIBC_2.0) [SUSv3]	getchar(GLIBC_2.0) [SUSv3]	getchar_unlocked(GLIBC_2.0) [SUSv3]
getw(GLIBC_2.0) [SUSv2]	pclose(GLIBC_2.1) [SUSv3]	popen(GLIBC_2.1) [SUSv3]	printf(GLIBC_2.0) [SUSv3]
putc(GLIBC_2.0) [SUSv3]	putc_unlocked(GLIBC_2.0) [SUSv3]	putchar(GLIBC_2.0) [SUSv3]	putchar_unlocked(GLIBC_2.0) [SUSv3]
puts(GLIBC_2.0) [SUSv3]	putw(GLIBC_2.0) [SUSv2]	remove(GLIBC_2.0) [SUSv3]	rewind(GLIBC_2.0) [SUSv3]
rewinddir(GLIBC_2.0) [SUSv3]	scanf(GLIBC_2.0) [LSB]	seekdir(GLIBC_2.0) [SUSv3]	setbuf(GLIBC_2.0) [SUSv3]
setbuffer(GLIBC_2.0) [LSB]	setvbuf(GLIBC_2.0) [SUSv3]	snprintf(GLIBC_2.0) [SUSv3]	sprintf(GLIBC_2.0) [SUSv3]
sscanf(GLIBC_2.0) [LSB]	telldir(GLIBC_2.0) [SUSv3]	tempnam(GLIBC_2.0) [SUSv3]	ungetc(GLIBC_2.0) [SUSv3]
vasprintf(GLIBC_2.0) [LSB]	vdprintf(GLIBC_2.0) [LSB]	vfprintf(GLIBC_2.0) [SUSv3]	vprintf(GLIBC_2.0) [SUSv3]
vsnprintf(GLIBC_2.0) [SUSv3]	vsprintf(GLIBC_2.0) [SUSv3]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard I/O specified in Table 11-5, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-5 libc - Standard I/O Data Interfaces

stderr(GLIBC_2.0) [SUSv3]	stdin(GLIBC_2.0) [SUSv3]	stdout(GLIBC_2.0) [SUSv3]	

11.2.4 Signal Handling

11.2.4.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the architecture specific functions for Signal Handling specified in Table 11-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-6 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.1) [LSB]	__libc_current_sigrtmin(GLIBC_2.1) [LSB]	__sigsetjmp(GLIBC_2.0) [LSB]	__sysv_signal(GLIBC_2.0) [LSB]
bsd_signal(GLIBC_2.0) [SUSv3]	psignal(GLIBC_2.0) [LSB]	raise(GLIBC_2.0) [SUSv3]	sigaction(GLIBC_2.0) [SUSv3]
sigaddset(GLIBC_2.0) [SUSv3]	sigaltstack(GLIBC_2.0) [SUSv3]	sigandset(GLIBC_2.0) [LSB]	sigdelset(GLIBC_2.0) [SUSv3]
sigemptyset(GLI	sigfillset(GLIBC_	sighold(GLIBC_2	sigignore(GLIBC

BC_2.0) [SUSv3]	2.0) [SUSv3]	.1) [SUSv3]	_2.1) [SUSv3]
siginterrupt(GLIBC_2.0) [SUSv3]	sigisemptyset(GLIBC_2.0) [LSB]	sigismember(GLIBC_2.0) [SUSv3]	siglongjmp(GLIBC_2.0) [SUSv3]
signal(GLIBC_2.0) [SUSv3]	sigorset(GLIBC_2.0) [LSB]	sigpause(GLIBC_2.0) [SUSv3]	sigpending(GLIBC_2.0) [SUSv3]
sigprocmask(GLIBC_2.0) [SUSv3]	sigqueue(GLIBC_2.1) [SUSv3]	sigrelse(GLIBC_2.1) [SUSv3]	sigreturn(GLIBC_2.0) [LSB]
sigset(GLIBC_2.1) [SUSv3]	sigsuspend(GLIBC_2.0) [SUSv3]	sigtimedwait(GLIBC_2.1) [SUSv3]	sigwait(GLIBC_2.0) [SUSv3]
sigwaitinfo(GLIBC_2.1) [SUSv3]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Signal Handling specified in Table 11-7, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-7 libc - Signal Handling Data Interfaces

_sys_siglist(GLIBC_2.3.3) [LSB]			

11.2.5 Localization Functions

11.2.5.1 Interfaces for Localization Functions

An LSB conforming implementation shall provide the architecture specific functions for Localization Functions specified in Table 11-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-8 libc - Localization Functions Function Interfaces

bind_textdomain_codeset(GLIBC_2.2) [LSB]	bindtextdomain(GLIBC_2.0) [LSB]	catclose(GLIBC_2.0) [SUSv3]	catgets(GLIBC_2.0) [SUSv3]
catopen(GLIBC_2.0) [SUSv3]	dcgettext(GLIBC_2.0) [LSB]	dcngettext(GLIBC_2.2) [LSB]	dgettext(GLIBC_2.0) [LSB]
dngettext(GLIBC_2.2) [LSB]	gettext(GLIBC_2.0) [LSB]	iconv(GLIBC_2.1) [SUSv3]	iconv_close(GLIBC_2.1) [SUSv3]
iconv_open(GLIBC_2.1) [SUSv3]	localeconv(GLIBC_2.2) [SUSv3]	ngettext(GLIBC_2.2) [LSB]	nl_langinfo(GLIBC_2.0) [SUSv3]
setlocale(GLIBC_2.0) [SUSv3]	textdomain(GLIBC_2.0) [LSB]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Localization Functions specified in Table 11-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-9 libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr (GLIBC_2.0)			

[LSB]			

11.2.6 Socket Interface

11.2.6.1 Interfaces for Socket Interface

An LSB conforming implementation shall provide the architecture specific functions for Socket Interface specified in Table 11-10, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-10 libc - Socket Interface Function Interfaces

__h_errno_location(GLIBC_2.0) [LSB]	accept(GLIBC_2.0) [SUSv3]	bind(GLIBC_2.0) [SUSv3]	bindresvport(GLIBC_2.0) [LSB]
connect(GLIBC_2.0) [SUSv3]	gethostid(GLIBC_2.0) [SUSv3]	gethostname(GLIBC_2.0) [SUSv3]	getpeername(GLIBC_2.0) [SUSv3]
getsockname(GLIBC_2.0) [SUSv3]	getsockopt(GLIBC_2.0) [LSB]	if_freenameindex(GLIBC_2.1) [SUSv3]	if_indextoname(GLIBC_2.1) [SUSv3]
if_nameindex(GLIBC_2.1) [SUSv3]	if_nametoindex(GLIBC_2.1) [SUSv3]	listen(GLIBC_2.0) [SUSv3]	recv(GLIBC_2.0) [SUSv3]
recvfrom(GLIBC_2.0) [SUSv3]	recvmsg(GLIBC_2.0) [SUSv3]	send(GLIBC_2.0) [SUSv3]	sendmsg(GLIBC_2.0) [SUSv3]
sendto(GLIBC_2.0) [SUSv3]	setsockopt(GLIBC_2.0) [LSB]	shutdown(GLIBC_2.0) [SUSv3]	sockatmark(GLIBC_2.2.4) [SUSv3]
socket(GLIBC_2.0) [SUSv3]	socketpair(GLIBC_2.0) [SUSv3]		

11.2.7 Wide Characters

11.2.7.1 Interfaces for Wide Characters

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

Table 11-11 libc - Wide Characters Function Interfaces

__wcstod_internal(GLIBC_2.0) [LSB]	__wcstof_internal(GLIBC_2.0) [LSB]	__wcstol_internal(GLIBC_2.0) [LSB]	__wcstold_internal(GLIBC_2.0) [LSB]
__wcstoul_internal(GLIBC_2.0) [LSB]	btowc(GLIBC_2.0) [SUSv3]	fgetwc(GLIBC_2.2) [SUSv3]	fgetws(GLIBC_2.2) [SUSv3]
fputwc(GLIBC_2.2) [SUSv3]	fputws(GLIBC_2.2) [SUSv3]	fwide(GLIBC_2.2) [SUSv3]	fwprintf(GLIBC_2.2) [SUSv3]
fwscanf(GLIBC_2.2) [LSB]	getwc(GLIBC_2.2) [SUSv3]	getwchar(GLIBC_2.2) [SUSv3]	mblen(GLIBC_2.0) [SUSv3]

mbrlen(GLIBC_2.0) [SUSv3]	mbrtowc(GLIBC_2.0) [SUSv3]	mbsinit(GLIBC_2.0) [SUSv3]	mbsnrtowcs(GLIBC_2.0) [LSB]
mbsrtowcs(GLIBC_2.0) [SUSv3]	mbstowcs(GLIBC_2.0) [SUSv3]	mbtowc(GLIBC_2.0) [SUSv3]	putwc(GLIBC_2.2) [SUSv3]
putwchar(GLIBC_2.2) [SUSv3]	swprintf(GLIBC_2.2) [SUSv3]	swscanf(GLIBC_2.2) [LSB]	towctrans(GLIBC_2.0) [SUSv3]
towlower(GLIBC_2.0) [SUSv3]	towupper(GLIBC_2.0) [SUSv3]	ungetwc(GLIBC_2.2) [SUSv3]	vfwprintf(GLIBC_2.2) [SUSv3]
vfwscanf(GLIBC_2.2) [LSB]	vswprintf(GLIBC_2.2) [SUSv3]	vswscanf(GLIBC_2.2) [LSB]	vwprintf(GLIBC_2.2) [SUSv3]
vwscanf(GLIBC_2.2) [LSB]	wcpcpy(GLIBC_2.0) [LSB]	wcpncpy(GLIBC_2.0) [LSB]	wcrtomb(GLIBC_2.0) [SUSv3]
wcscasecmp(GLIBC_2.1) [LSB]	wcscat(GLIBC_2.0) [SUSv3]	wcschr(GLIBC_2.0) [SUSv3]	wcscmp(GLIBC_2.0) [SUSv3]
wcscoll(GLIBC_2.0) [SUSv3]	wcscpy(GLIBC_2.0) [SUSv3]	wcscspn(GLIBC_2.0) [SUSv3]	wcsdup(GLIBC_2.0) [LSB]
wcsftime(GLIBC_2.2) [SUSv3]	wcslen(GLIBC_2.0) [SUSv3]	wcsncasecmp(GLIBC_2.1) [LSB]	wcsncat(GLIBC_2.0) [SUSv3]
wcsncmp(GLIBC_2.0) [SUSv3]	wcsncpy(GLIBC_2.0) [SUSv3]	wcsnlen(GLIBC_2.1) [LSB]	wcsnrtombs(GLIBC_2.0) [LSB]
wcspbrk(GLIBC_2.0) [SUSv3]	wcsrchr(GLIBC_2.0) [SUSv3]	wcsrtombs(GLIBC_2.0) [SUSv3]	wcsspn(GLIBC_2.0) [SUSv3]
wcsstr(GLIBC_2.0) [SUSv3]	wcstod(GLIBC_2.0) [SUSv3]	wcstof(GLIBC_2.0) [SUSv3]	wcstoimax(GLIBC_2.1) [SUSv3]
wcstok(GLIBC_2.0) [SUSv3]	wcstol(GLIBC_2.0) [SUSv3]	wcstold(GLIBC_2.0) [SUSv3]	wcstoll(GLIBC_2.1) [SUSv3]
wcstombs(GLIBC_2.0) [SUSv3]	wcstoq(GLIBC_2.0) [LSB]	wcstoul(GLIBC_2.0) [SUSv3]	wcstoull(GLIBC_2.1) [SUSv3]
wcstoumax(GLIBC_2.1) [SUSv3]	wcstouq(GLIBC_2.0) [LSB]	wcswcs(GLIBC_2.1) [SUSv3]	wcswidth(GLIBC_2.0) [SUSv3]
wcsxfrm(GLIBC_2.0) [SUSv3]	wctob(GLIBC_2.0) [SUSv3]	wctomb(GLIBC_2.0) [SUSv3]	wctrans(GLIBC_2.0) [SUSv3]
wctype(GLIBC_2.0) [SUSv3]	wcwidth(GLIBC_2.0) [SUSv3]	wmemchr(GLIBC_2.0) [SUSv3]	wmemcmp(GLIBC_2.0) [SUSv3]
wmemcpy(GLIBC_2.0) [SUSv3]	wmemmove(GLIBC_2.0) [SUSv3]	wmemset(GLIBC_2.0) [SUSv3]	wprintf(GLIBC_2.2) [SUSv3]
wscanf(GLIBC_2.2) [LSB]			

11.2.8 String Functions

11.2.8.1 Interfaces for String Functions

An LSB conforming implementation shall provide the architecture specific functions for String Functions specified in Table 11-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-12 libc - String Functions Function Interfaces

__mempcpy(GLIBC_2.0) [LSB]	__rawmemchr(GLIBC_2.1) [LSB]	__stpcpy(GLIBC_2.0) [LSB]	__strdup(GLIBC_2.0) [LSB]
__strtod_internal(GLIBC_2.0) [LSB]	__strtof_internal(GLIBC_2.0) [LSB]	__strtok_r(GLIBC_2.0) [LSB]	__strtol_internal(GLIBC_2.0) [LSB]
__strtold_internal(GLIBC_2.0) [LSB]	__strtoll_internal(GLIBC_2.0) [LSB]	__strtoul_internal(GLIBC_2.0) [LSB]	__strtoull_internal(GLIBC_2.0) [LSB]
bcmp(GLIBC_2.0) [SUSv3]	bcopy(GLIBC_2.0) [SUSv3]	bzero(GLIBC_2.0) [SUSv3]	ffs(GLIBC_2.0) [SUSv3]
index(GLIBC_2.0) [SUSv3]	memccpy(GLIBC_2.0) [SUSv3]	memchr(GLIBC_2.0) [SUSv3]	memcmp(GLIBC_2.0) [SUSv3]
memcpy(GLIBC_2.0) [SUSv3]	memmove(GLIBC_2.0) [SUSv3]	memrchr(GLIBC_2.2) [LSB]	memset(GLIBC_2.0) [SUSv3]
rindex(GLIBC_2.0) [SUSv3]	stpcpy(GLIBC_2.0) [LSB]	stpncpy(GLIBC_2.0) [LSB]	strcasecmp(GLIBC_2.0) [SUSv3]
strcasestr(GLIBC_2.1) [LSB]	strcat(GLIBC_2.0) [SUSv3]	strchr(GLIBC_2.0) [SUSv3]	strcmp(GLIBC_2.0) [SUSv3]
strcoll(GLIBC_2.0) [SUSv3]	strcpy(GLIBC_2.0) [SUSv3]	strcspn(GLIBC_2.0) [SUSv3]	strdup(GLIBC_2.0) [SUSv3]
strerror(GLIBC_2.0) [SUSv3]	strerror_r(GLIBC_2.0) [LSB]	strfmon(GLIBC_2.0) [SUSv3]	strftime(GLIBC_2.0) [SUSv3]
strlen(GLIBC_2.0) [SUSv3]	strncasecmp(GLIBC_2.0) [SUSv3]	strncat(GLIBC_2.0) [SUSv3]	strncmp(GLIBC_2.0) [SUSv3]
strncpy(GLIBC_2.0) [SUSv3]	strndup(GLIBC_2.0) [LSB]	strnlen(GLIBC_2.0) [LSB]	strpbrk(GLIBC_2.0) [SUSv3]
strptime(GLIBC_2.0) [LSB]	strrchr(GLIBC_2.0) [SUSv3]	strsep(GLIBC_2.0) [LSB]	strsignal(GLIBC_2.0) [LSB]
strspn(GLIBC_2.0) [SUSv3]	strstr(GLIBC_2.0) [SUSv3]	strtof(GLIBC_2.0) [SUSv3]	strtoimax(GLIBC_2.1) [SUSv3]
strtok(GLIBC_2.0) [SUSv3]	strtok_r(GLIBC_2.0) [SUSv3]	strtold(GLIBC_2.0) [SUSv3]	strtoll(GLIBC_2.0) [SUSv3]
strtoq(GLIBC_2.0) [LSB]	strtoull(GLIBC_2.0) [SUSv3]	strtoumax(GLIBC_2.1) [SUSv3]	strtouq(GLIBC_2.0) [LSB]
strxfrm(GLIBC_2.0) [SUSv3]	swab(GLIBC_2.0) [SUSv3]		

11.2.9 IPC Functions

11.2.9.1 Interfaces for IPC Functions

An LSB conforming implementation shall provide the architecture specific functions for IPC Functions specified in Table 11-13, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-13 libc - IPC Functions Function Interfaces

ftok(GLIBC_2.0) [SUSv3]	msgctl(GLIBC_2.2) [SUSv3]	msgget(GLIBC_2.0) [SUSv3]	msgrcv(GLIBC_2.0) [SUSv3]
msgsnd(GLIBC_2.0) [SUSv3]	semctl(GLIBC_2.2) [SUSv3]	semget(GLIBC_2.0) [SUSv3]	semop(GLIBC_2.0) [SUSv3]
shmat(GLIBC_2.0) [SUSv3]	shmctl(GLIBC_2.2) [SUSv3]	shmdt(GLIBC_2.0) [SUSv3]	shmget(GLIBC_2.0) [SUSv3]

11.2.10 Regular Expressions

11.2.10.1 Interfaces for Regular Expressions

An LSB conforming implementation shall provide the architecture specific functions for Regular Expressions specified in Table 11-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-14 libc - Regular Expressions Function Interfaces

regcomp(GLIBC_2.0) [SUSv3]	regerror(GLIBC_2.0) [SUSv3]	regexec(GLIBC_2.3.4) [LSB]	regfree(GLIBC_2.0) [SUSv3]

11.2.11 Character Type Functions

11.2.11.1 Interfaces for Character Type Functions

An LSB conforming implementation shall provide the architecture specific functions for Character Type Functions specified in Table 11-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-15 libc - Character Type Functions Function Interfaces

__ctype_get_mb_cur_max(GLIBC_2.0) [LSB]	_tolower(GLIBC_2.0) [SUSv3]	_toupper(GLIBC_2.0) [SUSv3]	isalnum(GLIBC_2.0) [SUSv3]
isalpha(GLIBC_2.0) [SUSv3]	isascii(GLIBC_2.0) [SUSv3]	iscntrl(GLIBC_2.0) [SUSv3]	isdigit(GLIBC_2.0) [SUSv3]
isgraph(GLIBC_2.0) [SUSv3]	islower(GLIBC_2.0) [SUSv3]	isprint(GLIBC_2.0) [SUSv3]	ispunct(GLIBC_2.0) [SUSv3]
isspace(GLIBC_2.0) [SUSv3]	isupper(GLIBC_2.0) [SUSv3]	iswalnum(GLIBC_2.0) [SUSv3]	iswalpha(GLIBC_2.0) [SUSv3]
iswblank(GLIBC_2.1) [SUSv3]	iswcntrl(GLIBC_2.0) [SUSv3]	iswctype(GLIBC_2.0) [SUSv3]	iswdigit(GLIBC_2.0) [SUSv3]
iswgraph(GLIBC_2.0) [SUSv3]	iswlower(GLIBC_2.0) [SUSv3]	iswprint(GLIBC_2.0) [SUSv3]	iswpunct(GLIBC_2.0) [SUSv3]

iswspace(GLIBC_2.0) [SUSv3]	iswupper(GLIBC_2.0) [SUSv3]	iswxdigit(GLIBC_2.0) [SUSv3]	isxdigit(GLIBC_2.0) [SUSv3]
toascii(GLIBC_2.0) [SUSv3]	tolower(GLIBC_2.0) [SUSv3]	toupper(GLIBC_2.0) [SUSv3]	

11.2.12 Time Manipulation

11.2.12.1 Interfaces for Time Manipulation

An LSB conforming implementation shall provide the architecture specific functions for Time Manipulation specified in Table 11-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-16 libc - Time Manipulation Function Interfaces

adjtime(GLIBC_2.0) [LSB]	asctime(GLIBC_2.0) [SUSv3]	asctime_r(GLIBC_2.0) [SUSv3]	ctime(GLIBC_2.0) [SUSv3]
ctime_r(GLIBC_2.0) [SUSv3]	difftime(GLIBC_2.0) [SUSv3]	gmtime(GLIBC_2.0) [SUSv3]	gmtime_r(GLIBC_2.0) [SUSv3]
localtime(GLIBC_2.0) [SUSv3]	localtime_r(GLIBC_2.0) [SUSv3]	mktime(GLIBC_2.0) [SUSv3]	tzset(GLIBC_2.0) [SUSv3]
ualarm(GLIBC_2.0) [SUSv3]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Time Manipulation specified in Table 11-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-17 libc - Time Manipulation Data Interfaces

__daylight(GLIBC_2.0) [LSB]	__timezone(GLIBC_2.0) [LSB]	__tzname(GLIBC_2.0) [LSB]	daylight(GLIBC_2.0) [SUSv3]
timezone(GLIBC_2.0) [SUSv3]	tzname(GLIBC_2.0) [SUSv3]		

11.2.13 Terminal Interface Functions

11.2.13.1 Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the architecture specific functions for Terminal Interface Functions specified in Table 11-18, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-18 libc - Terminal Interface Functions Function Interfaces

cfgetispeed(GLIBC_2.0) [SUSv3]	cfgetospeed(GLIBC_2.0) [SUSv3]	cfmakeraw(GLIBC_2.0) [LSB]	cfsetispeed(GLIBC_2.0) [SUSv3]
cfsetospeed(GLIBC_2.0) [SUSv3]	cfsetspeed(GLIBC_2.0) [LSB]	tcdrain(GLIBC_2.0) [SUSv3]	tcflow(GLIBC_2.0) [SUSv3]
tcflush(GLIBC_2.0) [SUSv3]	tcgetattr(GLIBC_2.0) [SUSv3]	tcgetpgrp(GLIBC_2.0) [SUSv3]	tcgetsid(GLIBC_2.1) [SUSv3]
tcsendbreak(GLI	tcsetattr(GLIBC_	tcsetpgrp(GLIBC	

BC_2.0) [SUSv3]	2.0) [SUSv3]	_2.0) [SUSv3]	

11.2.14 System Database Interface

11.2.14.1 Interfaces for System Database Interface

An LSB conforming implementation shall provide the architecture specific functions for System Database Interface specified in Table 11-19, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-19 libc - System Database Interface Function Interfaces

endgrent(GLIBC_2.0) [SUSv3]	endprotoent(GLIBC_2.0) [SUSv3]	endpwent(GLIBC_2.0) [SUSv3]	endservent(GLIBC_2.0) [SUSv3]
endutent(GLIBC_2.0) [LSB]	endutxent(GLIBC_2.1) [SUSv3]	getgrent(GLIBC_2.0) [SUSv3]	getgrgid(GLIBC_2.0) [SUSv3]
getgrgid_r(GLIBC_2.1.2) [SUSv3]	getgrnam(GLIBC_2.0) [SUSv3]	getgrnam_r(GLIBC_2.1.2) [SUSv3]	getgrouplist(GLIBC_2.2.4) [LSB]
gethostbyaddr(GLIBC_2.0) [SUSv3]	gethostbyname(GLIBC_2.0) [SUSv3]	getprotobyname(GLIBC_2.0) [SUSv3]	getprotobynumber(GLIBC_2.0) [SUSv3]
getprotoent(GLIBC_2.0) [SUSv3]	getpwent(GLIBC_2.0) [SUSv3]	getpwnam(GLIBC_2.0) [SUSv3]	getpwnam_r(GLIBC_2.1.2) [SUSv3]
getpwuid(GLIBC_2.0) [SUSv3]	getpwuid_r(GLIBC_2.1.2) [SUSv3]	getservbyname(GLIBC_2.0) [SUSv3]	getservbyport(GLIBC_2.0) [SUSv3]
getservent(GLIBC_2.0) [SUSv3]	getutent(GLIBC_2.0) [LSB]	getutent_r(GLIBC_2.0) [LSB]	getutxent(GLIBC_2.1) [SUSv3]
getutxid(GLIBC_2.1) [SUSv3]	getutxline(GLIBC_2.1) [SUSv3]	pututxline(GLIBC_2.1) [SUSv3]	setgrent(GLIBC_2.0) [SUSv3]
setgroups(GLIBC_2.0) [LSB]	setprotoent(GLIBC_2.0) [SUSv3]	setpwent(GLIBC_2.0) [SUSv3]	setservent(GLIBC_2.0) [SUSv3]
setutent(GLIBC_2.0) [LSB]	setutxent(GLIBC_2.1) [SUSv3]	utmpname(GLIBC_2.0) [LSB]	

11.2.15 Language Support

11.2.15.1 Interfaces for Language Support

An LSB conforming implementation shall provide the architecture specific functions for Language Support specified in Table 11-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-20 libc - Language Support Function Interfaces

__libc_start_main(GLIBC_2.0) [LSB]			

11.2.16 Large File Support

11.2.16.1 Interfaces for Large File Support

An LSB conforming implementation shall provide the architecture specific functions for Large File Support specified in Table 11-21, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-21 libc - Large File Support Function Interfaces

__fxstat64(GLIBC_2.2) [LSB]	__lxstat64(GLIBC_2.2) [LSB]	__xstat64(GLIBC_2.2) [LSB]	creat64(GLIBC_2.1) [LFS]
fgetpos64(GLIBC_2.2) [LFS]	fopen64(GLIBC_2.1) [LFS]	freopen64(GLIBC_2.1) [LFS]	fseeko64(GLIBC_2.1) [LFS]
fsetpos64(GLIBC_2.2) [LFS]	fstatvfs64(GLIBC_2.1) [LFS]	ftello64(GLIBC_2.1) [LFS]	ftruncate64(GLIBC_2.1) [LFS]
ftw64(GLIBC_2.1) [LFS]	getrlimit64(GLIBC_2.2) [LFS]	lockf64(GLIBC_2.1) [LFS]	mkstemp64(GLIBC_2.2) [LFS]
mmap64(GLIBC_2.1) [LFS]	nftw64(GLIBC_2.3.3) [LFS]	readdir64(GLIBC_2.2) [LFS]	statvfs64(GLIBC_2.1) [LFS]
tmpfile64(GLIBC_2.1) [LFS]	truncate64(GLIBC_2.1) [LFS]		

11.2.17 Standard Library

11.2.17.1 Interfaces for Standard Library

An LSB conforming implementation shall provide the architecture specific functions for Standard Library specified in Table 11-22, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-22 libc - Standard Library Function Interfaces

_Exit(GLIBC_2.1.1) [SUSv3]	__assert_fail(GLIBC_2.0) [LSB]	__cxa_atexit(GLIBC_2.1.3) [LSB]	__errno_location(GLIBC_2.0) [LSB]
__fpending(GLIBC_2.2) [LSB]	__getpagesize(GLIBC_2.0) [LSB]	__isinf(GLIBC_2.0) [LSB]	__isinff(GLIBC_2.0) [LSB]
__isinfl(GLIBC_2.0) [LSB]	__isnan(GLIBC_2.0) [LSB]	__isnanf(GLIBC_2.0) [LSB]	__isnanl(GLIBC_2.0) [LSB]
__sysconf(GLIBC_2.2) [LSB]	_exit(GLIBC_2.0) [SUSv3]	_longjmp(GLIBC_2.0) [SUSv3]	_setjmp(GLIBC_2.0) [SUSv3]
a64l(GLIBC_2.0) [SUSv3]	abort(GLIBC_2.0) [SUSv3]	abs(GLIBC_2.0) [SUSv3]	atof(GLIBC_2.0) [SUSv3]
atoi(GLIBC_2.0) [SUSv3]	atol(GLIBC_2.0) [SUSv3]	atoll(GLIBC_2.0) [SUSv3]	basename(GLIBC_2.0) [SUSv3]
bsearch(GLIBC_2.0) [SUSv3]	calloc(GLIBC_2.0) [SUSv3]	closelog(GLIBC_2.0) [SUSv3]	confstr(GLIBC_2.0) [SUSv3]
cuserid(GLIBC_2.0) [SUSv2]	daemon(GLIBC_2.0) [LSB]	dirname(GLIBC_2.0) [SUSv3]	div(GLIBC_2.0) [SUSv3]

drand48(GLIBC_2.0) [SUSv3]	ecvt(GLIBC_2.0) [SUSv3]	erand48(GLIBC_2.0) [SUSv3]	err(GLIBC_2.0) [LSB]
error(GLIBC_2.0) [LSB]	errx(GLIBC_2.0) [LSB]	fcvt(GLIBC_2.0) [SUSv3]	fmtmsg(GLIBC_2.1) [SUSv3]
fnmatch(GLIBC_2.2.3) [SUSv3]	fpathconf(GLIBC_2.0) [SUSv3]	free(GLIBC_2.0) [SUSv3]	freeaddrinfo(GLIBC_2.0) [SUSv3]
ftrylockfile(GLIBC_2.0) [SUSv3]	ftw(GLIBC_2.0) [SUSv3]	funlockfile(GLIBC_2.0) [SUSv3]	gai_strerror(GLIBC_2.1) [SUSv3]
gcvt(GLIBC_2.0) [SUSv3]	getaddrinfo(GLIBC_2.0) [SUSv3]	getcwd(GLIBC_2.0) [SUSv3]	getdate(GLIBC_2.1) [SUSv3]
getenv(GLIBC_2.0) [SUSv3]	getlogin(GLIBC_2.0) [SUSv3]	getlogin_r(GLIBC_2.0) [SUSv3]	getnameinfo(GLIBC_2.1) [SUSv3]
getopt(GLIBC_2.0) [LSB]	getopt_long(GLIBC_2.0) [LSB]	getopt_long_only(GLIBC_2.0) [LSB]	getsubopt(GLIBC_2.0) [SUSv3]
gettimeofday(GLIBC_2.0) [SUSv3]	glob(GLIBC_2.0) [SUSv3]	glob64(GLIBC_2.2) [LSB]	globfree(GLIBC_2.0) [SUSv3]
globfree64(GLIBC_2.1) [LSB]	grantpt(GLIBC_2.1) [SUSv3]	hcreate(GLIBC_2.0) [SUSv3]	hdestroy(GLIBC_2.0) [SUSv3]
hsearch(GLIBC_2.0) [SUSv3]	htonl(GLIBC_2.0) [SUSv3]	htons(GLIBC_2.0) [SUSv3]	imaxabs(GLIBC_2.1.1) [SUSv3]
imaxdiv(GLIBC_2.1.1) [SUSv3]	inet_addr(GLIBC_2.0) [SUSv3]	inet_ntoa(GLIBC_2.0) [SUSv3]	inet_ntop(GLIBC_2.0) [SUSv3]
inet_pton(GLIBC_2.0) [SUSv3]	initstate(GLIBC_2.0) [SUSv3]	insque(GLIBC_2.0) [SUSv3]	isatty(GLIBC_2.0) [SUSv3]
isblank(GLIBC_2.0) [SUSv3]	jrand48(GLIBC_2.0) [SUSv3]	l64a(GLIBC_2.0) [SUSv3]	labs(GLIBC_2.0) [SUSv3]
lcong48(GLIBC_2.0) [SUSv3]	ldiv(GLIBC_2.0) [SUSv3]	lfind(GLIBC_2.0) [SUSv3]	llabs(GLIBC_2.0) [SUSv3]
lldiv(GLIBC_2.0) [SUSv3]	longjmp(GLIBC_2.0) [SUSv3]	lrand48(GLIBC_2.0) [SUSv3]	lsearch(GLIBC_2.0) [SUSv3]
makecontext(GLIBC_2.1) [SUSv3]	malloc(GLIBC_2.0) [SUSv3]	memmem(GLIBC_2.0) [LSB]	mkstemp(GLIBC_2.0) [SUSv3]
mktemp(GLIBC_2.0) [SUSv3]	mrand48(GLIBC_2.0) [SUSv3]	nftw(GLIBC_2.3.3) [SUSv3]	nrand48(GLIBC_2.0) [SUSv3]
ntohl(GLIBC_2.0) [SUSv3]	ntohs(GLIBC_2.0) [SUSv3]	openlog(GLIBC_2.0) [SUSv3]	perror(GLIBC_2.0) [SUSv3]
posix_memalign(GLIBC_2.2) [SUSv3]	posix_openpt(GLIBC_2.2.1) [SUSv3]	ptsname(GLIBC_2.1) [SUSv3]	putenv(GLIBC_2.0) [SUSv3]
qsort(GLIBC_2.0) [SUSv3]	rand(GLIBC_2.0) [SUSv3]	rand_r(GLIBC_2.0) [SUSv3]	random(GLIBC_2.0) [SUSv3]
realloc(GLIBC_2.	realpath(GLIBC_	remque(GLIBC_2	seed48(GLIBC_2.

0) [SUSv3]	2.3) [SUSv3]	.0) [SUSv3]	0) [SUSv3]
setenv(GLIBC_2.0) [SUSv3]	sethostname(GLIBC_2.0) [LSB]	setlogmask(GLIBC_2.0) [SUSv3]	setstate(GLIBC_2.0) [SUSv3]
srand(GLIBC_2.0) [SUSv3]	srand48(GLIBC_2.0) [SUSv3]	srandom(GLIBC_2.0) [SUSv3]	strtod(GLIBC_2.0) [SUSv3]
strtol(GLIBC_2.0) [SUSv3]	strtoul(GLIBC_2.0) [SUSv3]	swapcontext(GLIBC_2.1) [SUSv3]	syslog(GLIBC_2.0) [SUSv3]
system(GLIBC_2.0) [LSB]	tdelete(GLIBC_2.0) [SUSv3]	tfind(GLIBC_2.0) [SUSv3]	tmpfile(GLIBC_2.1) [SUSv3]
tmpnam(GLIBC_2.0) [SUSv3]	tsearch(GLIBC_2.0) [SUSv3]	ttyname(GLIBC_2.0) [SUSv3]	ttyname_r(GLIBC_2.0) [SUSv3]
twalk(GLIBC_2.0) [SUSv3]	unlockpt(GLIBC_2.1) [SUSv3]	unsetenv(GLIBC_2.0) [SUSv3]	usleep(GLIBC_2.0) [SUSv3]
verrx(GLIBC_2.0) [LSB]	vfscanf(GLIBC_2.0) [LSB]	vscanf(GLIBC_2.0) [LSB]	vsscanf(GLIBC_2.0) [LSB]
vsyslog(GLIBC_2.0) [LSB]	warn(GLIBC_2.0) [LSB]	warnx(GLIBC_2.0) [LSB]	wordexp(GLIBC_2.1) [SUSv3]
wordfree(GLIBC_2.1) [SUSv3]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard Library specified in Table 11-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-23 libc - Standard Library Data Interfaces

__environ(GLIBC_2.0) [LSB]	_environ(GLIBC_2.0) [LSB]	_sys_errlist(GLIBC_2.3) [LSB]	environ(GLIBC_2.0) [SUSv3]
getdate_err(GLIBC_2.1) [SUSv3]	optarg(GLIBC_2.0) [SUSv3]	opterr(GLIBC_2.0) [SUSv3]	optind(GLIBC_2.0) [SUSv3]
optopt(GLIBC_2.0) [SUSv3]			

11.3 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.

11.3.1 ctype.h

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

11.3.2 dirent.h

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

11.3.3 errno.h

```
#define EDEADLOCK       EDEADLK
```

11.3.4 fcntl.h

```
#define F_GETLK64       12
#define F_SETLK64       13
#define F_SETLKW64      14
```

11.3.5 fnmatch.h

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

11.3.6 ftw.h

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

11.3.7 getopt.h

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

11.3.8 glob.h

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

11.3.9 iconv.h

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

11.3.10 inttypes.h

```
typedef long long int intmax_t;
typedef unsigned int uintptr_t;
typedef unsigned long long int uintmax_t;
typedef unsigned long long int uint64_t;
```

11.3.11 langinfo.h

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

11.3.12 limits.h

```
#define LONG_MAX        0x7FFFFFFFL
#define ULONG_MAX       0xFFFFFFFFUL

#define CHAR_MAX        SCHAR_MAX
#define CHAR_MIN        SCHAR_MIN

#define PTHREAD_STACK_MIN     16384
```

11.3.13 locale.h

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

11.3.14 net/if.h

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

11.3.15 netdb.h

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

11.3.16 netinet/in.h

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



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

11.3.17 netinet/ip.h

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

11.3.18 netinet/tcp.h

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

11.3.19 netinet/udp.h

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

11.3.20 nl_types.h

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

11.3.21 pwd.h

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

11.3.22 regex.h

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

11.3.23 rpc/auth.h

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

11.3.24 rpc/clnt.h

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

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

11.3.25 rpc/rpc_msg.h

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

11.3.26 rpc/svc.h

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

11.3.27 rpc/types.h

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

11.3.28 rpc/xdr.h

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

11.3.29 sched.h

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

11.3.30 search.h

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

11.3.31 setjmp.h

```
typedef int __jmp_buf[6];
```

11.3.32 signal.h

```
#define SIGEV_PAD_SIZE  ((SIGEV_MAX_SIZE/sizeof(int))-3)

#define SI_PAD_SIZE     ((SI_MAX_SIZE/sizeof(int))-3)

struct sigaction {
```



```
    union {
        sighandler_t _sa_handler;
        void (*_sa_sigaction) (int, siginfo_t *, void *);
    } __sigaction_handler;
    sigset_t sa_mask;
    unsigned long int sa_flags;
    void (*sa_restorer) (void);
};

#define MINSIGSTKSZ     2048
#define SIGSTKSZ        8192

struct _fpreg {
    unsigned short significand[4];
    unsigned short exponent;
};
struct _fpxreg {
    unsigned short significand[4];
    unsigned short exponent;
    unsigned short padding[3];
};
struct _xmmreg {
    unsigned long int element[4];
};

struct _fpstate {
    unsigned long int cw;
    unsigned long int sw;
    unsigned long int tag;
    unsigned long int ipoff;
    unsigned long int cssel;
    unsigned long int dataoff;
    unsigned long int datasel;
    struct _fpreg _st[8];
    unsigned short status;
    unsigned short magic;
    unsigned long int _fxsr_env[6];
    unsigned long int mxcsr;
    unsigned long int reserved;
    struct _fpxreg _fxsr_st[8];
    struct _xmmreg _xmm[8];
    unsigned long int padding[56];
};

struct sigcontext {
    unsigned short gs;
    unsigned short __gsh;
    unsigned short fs;
    unsigned short __fsh;
    unsigned short es;
    unsigned short __esh;
    unsigned short ds;
    unsigned short __dsh;
    unsigned long int edi;
    unsigned long int esi;
    unsigned long int ebp;
    unsigned long int esp;
    unsigned long int ebx;
    unsigned long int edx;
    unsigned long int ecx;
    unsigned long int eax;
    unsigned long int trapno;
    unsigned long int err;
    unsigned long int eip;
    unsigned short cs;
    unsigned short __csh;
```

```
    unsigned long int eflags;
    unsigned long int esp_at_signal;
    unsigned short ss;
    unsigned short __ssh;
    struct _fpstate *fpstate;
    unsigned long int oldmask;
    unsigned long int cr2;
};
```

11.3.33 stddef.h

```
typedef unsigned int size_t;
typedef int ptrdiff_t;
```

11.3.34 stdio.h

```
#define __IO_FILE_SIZE  148
```

11.3.35 stdlib.h

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

11.3.36 sys/file.h

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

11.3.37 sys/ioctl.h

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

11.3.38 sys/ipc.h

```
struct ipc_perm {
    key_t __key;
    uid_t uid;
    gid_t gid;
    uid_t cuid;
    gid_t cgid;
    unsigned short mode;
    unsigned short __pad1;
    unsigned short __seq;
    unsigned short __pad2;
    unsigned long int __unused1;
    unsigned long int __unused2;
};
```

11.3.39 sys/mman.h

```
#define MCL_CURRENT     1
```



```
#define MCL_FUTURE      2
```

11.3.40 sys/msg.h

```
typedef unsigned long int msgqnum_t;
typedef unsigned long int msglen_t;

struct msqid_ds {
    struct ipc_perm msg_perm;
    time_t msg_stime;
    unsigned long int __unused1;
    time_t msg_rtime;
    unsigned long int __unused2;
    time_t msg_ctime;
    unsigned long int __unused3;
    unsigned long int __msg_cbytes;
    msgqnum_t msg_qnum;
    msglen_t msg_qbytes;
    pid_t msg_lspid;
    pid_t msg_lrpid;
    unsigned long int __unused4;
    unsigned long int __unused5;
};
```

11.3.41 sys/param.h

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

11.3.42 sys/poll.h

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

11.3.43 sys/resource.h

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

11.3.44 sys/sem.h

```
struct semid_ds {
    struct ipc_perm sem_perm;
    time_t sem_otime;
    unsigned long int __unused1;
    time_t sem_ctime;
    unsigned long int __unused2;
    unsigned long int sem_nsems;
    unsigned long int __unused3;
    unsigned long int __unused4;
};
```

11.3.45 sys/shm.h

```
#define SHMLBA  (__getpagesize())

typedef unsigned long int shmatt_t;

struct shmid_ds {
    struct ipc_perm shm_perm;
    int shm_segsz;
    time_t shm_atime;
    unsigned long int __unused1;
    time_t shm_dtime;
    unsigned long int __unused2;
    time_t shm_ctime;
    unsigned long int __unused3;
    pid_t shm_cpid;
    pid_t shm_lpid;
    shmatt_t shm_nattch;
    unsigned long int __unused4;
    unsigned long int __unused5;
};
```

11.3.46 sys/socket.h

```
typedef uint32_t __ss_aligntype;

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

11.3.47 sys/stat.h

```
#define _STAT_VER       3

struct stat {
    dev_t st_dev;
    unsigned short __pad1;
    unsigned long int st_ino;
    mode_t st_mode;
    nlink_t st_nlink;
    pid_t st_uid;
    gid_t st_gid;
    dev_t st_rdev;
    unsigned short __pad2;
    off_t st_size;
    blksize_t st_blksize;
    blkcnt_t st_blocks;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    unsigned long int __unused4;
    unsigned long int __unused5;
};
struct stat64 {
    dev_t st_dev;
    unsigned int __pad1;
    ino_t __st_ino;
    mode_t st_mode;
    nlink_t st_nlink;
    uid_t st_uid;
    gid_t st_gid;
```



```
    dev_t st_rdev;
    unsigned int __pad2;
    off64_t st_size;
    blksize_t st_blksize;
    blkcnt64_t st_blocks;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    ino64_t st_ino;
};
```

11.3.48 sys/statvfs.h

```
struct statvfs {
    unsigned long int f_bsize;
    unsigned long int f_frsize;
    fsblkcnt_t f_blocks;
    fsblkcnt_t f_bfree;
    fsblkcnt_t f_bavail;
    fsfilcnt_t f_files;
    fsfilcnt_t f_ffree;
    fsfilcnt_t f_favail;
    unsigned long int f_fsid;
    int __f_unused;
    unsigned long int f_flag;
    unsigned long int f_namemax;
    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;
    int __f_unused;
    unsigned long int f_flag;
    unsigned long int f_namemax;
    int __f_spare[6];
};
```

11.3.49 sys/time.h

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

11.3.50 sys/timeb.h

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

11.3.51 sys/times.h

```
/*
```

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

11.3.52 sys/types.h

```
typedef long long int int64_t;

typedef int32_t ssize_t;

#define __FDSET_LONGS   32
```

11.3.53 sys/un.h

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

11.3.54 sys/utsname.h

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

11.3.55 sys/wait.h

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

11.3.56 syslog.h

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

11.3.57 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
```

11.3.58 ucontext.h

```
typedef int greg_t;

#define NGREG   19

typedef greg_t gregset_t[19];

struct _libc_fpreg {
    unsigned short significand[4];
    unsigned short exponent;
};

struct _libc_fpstate {
    unsigned long int cw;
    unsigned long int sw;
    unsigned long int tag;
    unsigned long int ipoff;
    unsigned long int cssel;
    unsigned long int dataoff;
    unsigned long int datasel;
```

```
    struct _libc_fpreg _st[8];
    unsigned long int status;
};
typedef struct _libc_fpstate *fpregset_t;

typedef struct {
    gregset_t gregs;
    fpregset_t fpregs;
    unsigned long int oldmask;
    unsigned long int cr2;
} mcontext_t;

typedef struct ucontext {
    unsigned long int uc_flags;
    struct ucontext *uc_link;
    stack_t uc_stack;
    mcontext_t uc_mcontext;
    sigset_t uc_sigmask;
    struct _libc_fpstate __fpregs_mem;
} ucontext_t;
```

11.3.59 ulimit.h

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

11.3.60 unistd.h

```
typedef int intptr_t;
```

11.3.61 utime.h

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

11.3.62 utmp.h

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

struct utmp {
    short ut_type;
    pid_t ut_pid;
    char ut_line[UT_LINESIZE];
    char ut_id[4];
    char ut_user[UT_NAMESIZE];
    char ut_host[UT_HOSTSIZE];
    struct exit_status ut_exit;
    long int ut_session;
    struct timeval ut_tv;
    int32_t ut_addr_v6[4];
    char __unused[20];
};
```


11.3.63 utmpx.h

```
struct utmpx {
    short ut_type;
    pid_t ut_pid;
    char ut_line[UT_LINESIZE];
    char ut_id[4];
    char ut_user[UT_NAMESIZE];
    char ut_host[UT_HOSTSIZE];
    struct exit_status ut_exit;
    long int ut_session;
    struct timeval ut_tv;
    int32_t ut_addr_v6[4];
    char __unused[20];
};
```

11.3.64 wctype.h

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

11.3.65 wordexp.h

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

11.4 Interfaces for libm

Table 11-24 defines the library name and shared object name for the libm library

Table 11-24 libm Definition

Library:	libm
SONAME:	libm.so.6

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

[ISOC99] ISO C (1999)
[LSB] ISO/IEC 23360-1
[SUSv2] SUSv2
[SUSv3] ISO POSIX (2003)
[SVID.3] SVID Issue 3

11.4.1 Math

11.4.1.1 Interfaces for Math

An LSB conforming implementation shall provide the architecture specific functions for Math specified in Table 11-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-25 libm - Math Function Interfaces

__finite(GLIBC_2.1) [ISOC99]	__finitef(GLIBC_2.1) [ISOC99]	__finitel(GLIBC_2.1) [ISOC99]	__fpclassify(GLIBC_2.1) [LSB]
__fpclassifyf(GLIBC_2.1) [LSB]	__fpclassifyl(GLIBC_2.1) [LSB]	__signbit(GLIBC_2.1) [ISOC99]	__signbitf(GLIBC_2.1) [ISOC99]
__signbitl(GLIBC_2.1) [ISOC99]	acos(GLIBC_2.0) [SUSv3]	acosf(GLIBC_2.0) [SUSv3]	acosh(GLIBC_2.0) [SUSv3]
acoshf(GLIBC_2.0) [SUSv3]	acoshl(GLIBC_2.0) [SUSv3]	acosl(GLIBC_2.0) [SUSv3]	asin(GLIBC_2.0) [SUSv3]
asinf(GLIBC_2.0) [SUSv3]	asinh(GLIBC_2.0) [SUSv3]	asinhf(GLIBC_2.0) [SUSv3]	asinhl(GLIBC_2.0) [SUSv3]
asinl(GLIBC_2.0) [SUSv3]	atan(GLIBC_2.0) [SUSv3]	atan2(GLIBC_2.0) [SUSv3]	atan2f(GLIBC_2.0) [SUSv3]
atan2l(GLIBC_2.0) [SUSv3]	atanf(GLIBC_2.0) [SUSv3]	atanh(GLIBC_2.0) [SUSv3]	atanhf(GLIBC_2.0) [SUSv3]
atanhl(GLIBC_2.0) [SUSv3]	atanl(GLIBC_2.0) [SUSv3]	cabs(GLIBC_2.1) [SUSv3]	cabsf(GLIBC_2.1) [SUSv3]
cabsl(GLIBC_2.1) [SUSv3]	cacos(GLIBC_2.1) [SUSv3]	cacosf(GLIBC_2.1) [SUSv3]	cacosh(GLIBC_2.1) [SUSv3]
cacoshf(GLIBC_2.1) [SUSv3]	cacoshl(GLIBC_2.1) [SUSv3]	cacosl(GLIBC_2.1) [SUSv3]	carg(GLIBC_2.1) [SUSv3]
cargf(GLIBC_2.1) [SUSv3]	cargl(GLIBC_2.1) [SUSv3]	casin(GLIBC_2.1) [SUSv3]	casinf(GLIBC_2.1) [SUSv3]
casinh(GLIBC_2.1) [SUSv3]	casinhf(GLIBC_2.1) [SUSv3]	casinhl(GLIBC_2.1) [SUSv3]	casinl(GLIBC_2.1) [SUSv3]
catan(GLIBC_2.1) [SUSv3]	catanf(GLIBC_2.1) [SUSv3]	catanh(GLIBC_2.1) [SUSv3]	catanhf(GLIBC_2.1) [SUSv3]
catanhl(GLIBC_2.1) [SUSv3]	catanl(GLIBC_2.1) [SUSv3]	cbrt(GLIBC_2.0) [SUSv3]	cbrtf(GLIBC_2.0) [SUSv3]
cbrtl(GLIBC_2.0) [SUSv3]	ccos(GLIBC_2.1) [SUSv3]	ccosf(GLIBC_2.1) [SUSv3]	ccosh(GLIBC_2.1) [SUSv3]
ccoshf(GLIBC_2.1) [SUSv3]	ccoshl(GLIBC_2.1) [SUSv3]	ccosl(GLIBC_2.1) [SUSv3]	ceil(GLIBC_2.0) [SUSv3]
ceilf(GLIBC_2.0) [SUSv3]	ceill(GLIBC_2.0) [SUSv3]	cexp(GLIBC_2.1) [SUSv3]	cexpf(GLIBC_2.1) [SUSv3]
cexpl(GLIBC_2.1) [SUSv3]	cimag(GLIBC_2.1) [SUSv3]	cimagf(GLIBC_2.1) [SUSv3]	cimagl(GLIBC_2.1) [SUSv3]
clog(GLIBC_2.1) [SUSv3]	clog10(GLIBC_2.1) [ISOC99]	clog10f(GLIBC_2.1) [ISOC99]	clog10l(GLIBC_2.1) [ISOC99]
clogf(GLIBC_2.1) [SUSv3]	clogl(GLIBC_2.1) [SUSv3]	conj(GLIBC_2.1) [SUSv3]	conjf(GLIBC_2.1) [SUSv3]
conjl(GLIBC_2.1)	copysign(GLIBC	copysignf(GLIBC	copysignl(GLIBC

[SUSv3]	_2.0) [SUSv3]	_2.0) [SUSv3]	_2.0) [SUSv3]
cos(GLIBC_2.0) [SUSv3]	cosf(GLIBC_2.0) [SUSv3]	cosh(GLIBC_2.0) [SUSv3]	coshf(GLIBC_2.0) [SUSv3]
coshl(GLIBC_2.0) [SUSv3]	cosl(GLIBC_2.0) [SUSv3]	cpow(GLIBC_2.1) [SUSv3]	cpowf(GLIBC_2.1) [SUSv3]
cpowl(GLIBC_2.1) [SUSv3]	cproj(GLIBC_2.1) [SUSv3]	cprojf(GLIBC_2.1) [SUSv3]	cprojl(GLIBC_2.1) [SUSv3]
creal(GLIBC_2.1) [SUSv3]	crealf(GLIBC_2.1) [SUSv3]	creall(GLIBC_2.1) [SUSv3]	csin(GLIBC_2.1) [SUSv3]
csinf(GLIBC_2.1) [SUSv3]	csinh(GLIBC_2.1) [SUSv3]	csinhf(GLIBC_2.1) [SUSv3]	csinhl(GLIBC_2.1) [SUSv3]
csinl(GLIBC_2.1) [SUSv3]	csqrt(GLIBC_2.1) [SUSv3]	csqrtf(GLIBC_2.1) [SUSv3]	csqrtl(GLIBC_2.1) [SUSv3]
ctan(GLIBC_2.1) [SUSv3]	ctanf(GLIBC_2.1) [SUSv3]	ctanh(GLIBC_2.1) [SUSv3]	ctanhf(GLIBC_2.1) [SUSv3]
ctanhl(GLIBC_2.1) [SUSv3]	ctanl(GLIBC_2.1) [SUSv3]	dremf(GLIBC_2.0) [ISOC99]	dreml(GLIBC_2.0) [ISOC99]
erf(GLIBC_2.0) [SUSv3]	erfc(GLIBC_2.0) [SUSv3]	erfcf(GLIBC_2.0) [SUSv3]	erfcl(GLIBC_2.0) [SUSv3]
erff(GLIBC_2.0) [SUSv3]	erfl(GLIBC_2.0) [SUSv3]	exp(GLIBC_2.0) [SUSv3]	exp2(GLIBC_2.1) [SUSv3]
exp2f(GLIBC_2.1) [SUSv3]	exp2l(GLIBC_2.1) [SUSv3]	expf(GLIBC_2.0) [SUSv3]	expl(GLIBC_2.0) [SUSv3]
expm1(GLIBC_2.0) [SUSv3]	expm1f(GLIBC_2.0) [SUSv3]	expm1l(GLIBC_2.0) [SUSv3]	fabs(GLIBC_2.0) [SUSv3]
fabsf(GLIBC_2.0) [SUSv3]	fabsl(GLIBC_2.0) [SUSv3]	fdim(GLIBC_2.1) [SUSv3]	fdimf(GLIBC_2.1) [SUSv3]
fdiml(GLIBC_2.1) [SUSv3]	feclearexcept(GLIBC_2.2) [SUSv3]	fegetenv(GLIBC_2.2) [SUSv3]	fegetexceptflag(GLIBC_2.2) [SUSv3]
fegetround(GLIBC_2.1) [SUSv3]	feholdexcept(GLIBC_2.1) [SUSv3]	feraiseexcept(GLIBC_2.2) [SUSv3]	fesetenv(GLIBC_2.2) [SUSv3]
fesetexceptflag(GLIBC_2.2) [SUSv3]	fesetround(GLIBC_2.1) [SUSv3]	fetestexcept(GLIBC_2.1) [SUSv3]	feupdateenv(GLIBC_2.2) [SUSv3]
finite(GLIBC_2.0) [SUSv2]	finitef(GLIBC_2.0) [ISOC99]	finitel(GLIBC_2.0) [ISOC99]	floor(GLIBC_2.0) [SUSv3]
floorf(GLIBC_2.0) [SUSv3]	floorl(GLIBC_2.0) [SUSv3]	fma(GLIBC_2.1) [SUSv3]	fmaf(GLIBC_2.1) [SUSv3]
fmal(GLIBC_2.1) [SUSv3]	fmax(GLIBC_2.1) [SUSv3]	fmaxf(GLIBC_2.1) [SUSv3]	fmaxl(GLIBC_2.1) [SUSv3]
fmin(GLIBC_2.1) [SUSv3]	fminf(GLIBC_2.1) [SUSv3]	fminl(GLIBC_2.1) [SUSv3]	fmod(GLIBC_2.0) [SUSv3]

fmodf(GLIBC_2.0) [SUSv3]	fmodl(GLIBC_2.0) [SUSv3]	frexp(GLIBC_2.0) [SUSv3]	frexpf(GLIBC_2.0) [SUSv3]
frexpl(GLIBC_2.0) [SUSv3]	gamma(GLIBC_2.0) [SUSv2]	gammaf(GLIBC_2.0) [ISOC99]	gammal(GLIBC_2.0) [ISOC99]
hypot(GLIBC_2.0) [SUSv3]	hypotf(GLIBC_2.0) [SUSv3]	hypotl(GLIBC_2.0) [SUSv3]	ilogb(GLIBC_2.0) [SUSv3]
ilogbf(GLIBC_2.0) [SUSv3]	ilogbl(GLIBC_2.0) [SUSv3]	j0(GLIBC_2.0) [SUSv3]	j0f(GLIBC_2.0) [ISOC99]
j0l(GLIBC_2.0) [ISOC99]	j1(GLIBC_2.0) [SUSv3]	j1f(GLIBC_2.0) [ISOC99]	j1l(GLIBC_2.0) [ISOC99]
jn(GLIBC_2.0) [SUSv3]	jnf(GLIBC_2.0) [ISOC99]	jnl(GLIBC_2.0) [ISOC99]	ldexp(GLIBC_2.0) [SUSv3]
ldexpf(GLIBC_2.0) [SUSv3]	ldexpl(GLIBC_2.0) [SUSv3]	lgamma(GLIBC_2.0) [SUSv3]	lgamma_r(GLIBC_2.0) [ISOC99]
lgammaf(GLIBC_2.0) [SUSv3]	lgammaf_r(GLIBC_2.0) [ISOC99]	lgammal(GLIBC_2.0) [SUSv3]	lgammal_r(GLIBC_2.0) [ISOC99]
llrint(GLIBC_2.1) [SUSv3]	llrintf(GLIBC_2.1) [SUSv3]	llrintl(GLIBC_2.1) [SUSv3]	llround(GLIBC_2.1) [SUSv3]
llroundf(GLIBC_2.1) [SUSv3]	llroundl(GLIBC_2.1) [SUSv3]	log(GLIBC_2.0) [SUSv3]	log10(GLIBC_2.0) [SUSv3]
log10f(GLIBC_2.0) [SUSv3]	log10l(GLIBC_2.0) [SUSv3]	log1p(GLIBC_2.0) [SUSv3]	log1pf(GLIBC_2.0) [SUSv3]
log1pl(GLIBC_2.0) [SUSv3]	log2(GLIBC_2.1) [SUSv3]	log2f(GLIBC_2.1) [SUSv3]	log2l(GLIBC_2.1) [SUSv3]
logb(GLIBC_2.0) [SUSv3]	logbf(GLIBC_2.0) [SUSv3]	logbl(GLIBC_2.0) [SUSv3]	logf(GLIBC_2.0) [SUSv3]
logl(GLIBC_2.0) [SUSv3]	lrint(GLIBC_2.1) [SUSv3]	lrintf(GLIBC_2.1) [SUSv3]	lrintl(GLIBC_2.1) [SUSv3]
lround(GLIBC_2.1) [SUSv3]	lroundf(GLIBC_2.1) [SUSv3]	lroundl(GLIBC_2.1) [SUSv3]	matherr(GLIBC_2.0) [SVID.3]
modf(GLIBC_2.0) [SUSv3]	modff(GLIBC_2.0) [SUSv3]	modfl(GLIBC_2.0) [SUSv3]	nan(GLIBC_2.1) [SUSv3]
nanf(GLIBC_2.1) [SUSv3]	nanl(GLIBC_2.1) [SUSv3]	nearbyint(GLIBC_2.1) [SUSv3]	nearbyintf(GLIBC_2.1) [SUSv3]
nearbyintl(GLIBC_2.1) [SUSv3]	nextafter(GLIBC_2.0) [SUSv3]	nextafterf(GLIBC_2.0) [SUSv3]	nextafterl(GLIBC_2.0) [SUSv3]
nexttoward(GLIBC_2.1) [SUSv3]	nexttowardf(GLIBC_2.1) [SUSv3]	nexttowardl(GLIBC_2.1) [SUSv3]	pow(GLIBC_2.0) [SUSv3]
pow10(GLIBC_2.1) [ISOC99]	pow10f(GLIBC_2.1) [ISOC99]	pow10l(GLIBC_2.1) [ISOC99]	powf(GLIBC_2.0) [SUSv3]
powl(GLIBC_2.0) [SUSv3]	remainder(GLIBC_2.0) [SUSv3]	remainderf(GLIBC_2.0) [SUSv3]	remainderl(GLIBC_2.0) [SUSv3]

remquo(GLIBC_2.1) [SUSv3]	remquof(GLIBC_2.1) [SUSv3]	remquol(GLIBC_2.1) [SUSv3]	rint(GLIBC_2.0) [SUSv3]
rintf(GLIBC_2.0) [SUSv3]	rintl(GLIBC_2.0) [SUSv3]	round(GLIBC_2.1) [SUSv3]	roundf(GLIBC_2.1) [SUSv3]
roundl(GLIBC_2.1) [SUSv3]	scalb(GLIBC_2.0) [SUSv3]	scalbf(GLIBC_2.0) [ISOC99]	scalbl(GLIBC_2.0) [ISOC99]
scalbln(GLIBC_2.1) [SUSv3]	scalblnf(GLIBC_2.1) [SUSv3]	scalblnl(GLIBC_2.1) [SUSv3]	scalbn(GLIBC_2.0) [SUSv3]
scalbnf(GLIBC_2.0) [SUSv3]	scalbnl(GLIBC_2.0) [SUSv3]	significand(GLIBC_2.0) [ISOC99]	significandf(GLIBC_2.0) [ISOC99]
significandl(GLIBC_2.0) [ISOC99]	sin(GLIBC_2.0) [SUSv3]	sincos(GLIBC_2.1) [ISOC99]	sincosf(GLIBC_2.1) [ISOC99]
sincosl(GLIBC_2.1) [ISOC99]	sinf(GLIBC_2.0) [SUSv3]	sinh(GLIBC_2.0) [SUSv3]	sinhf(GLIBC_2.0) [SUSv3]
sinhl(GLIBC_2.0) [SUSv3]	sinl(GLIBC_2.0) [SUSv3]	sqrt(GLIBC_2.0) [SUSv3]	sqrtf(GLIBC_2.0) [SUSv3]
sqrtl(GLIBC_2.0) [SUSv3]	tan(GLIBC_2.0) [SUSv3]	tanf(GLIBC_2.0) [SUSv3]	tanh(GLIBC_2.0) [SUSv3]
tanhf(GLIBC_2.0) [SUSv3]	tanhl(GLIBC_2.0) [SUSv3]	tanl(GLIBC_2.0) [SUSv3]	tgamma(GLIBC_2.1) [SUSv3]
tgammaf(GLIBC_2.1) [SUSv3]	tgammal(GLIBC_2.1) [SUSv3]	trunc(GLIBC_2.1) [SUSv3]	truncf(GLIBC_2.1) [SUSv3]
truncl(GLIBC_2.1) [SUSv3]	y0(GLIBC_2.0) [SUSv3]	y0f(GLIBC_2.0) [ISOC99]	y0l(GLIBC_2.0) [ISOC99]
y1(GLIBC_2.0) [SUSv3]	y1f(GLIBC_2.0) [ISOC99]	y1l(GLIBC_2.0) [ISOC99]	yn(GLIBC_2.0) [SUSv3]
ynf(GLIBC_2.0) [ISOC99]	ynl(GLIBC_2.0) [ISOC99]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Math specified in Table 11-26, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-26 libm - Math Data Interfaces

signgam(GLIBC_2.0) [SUSv3]			

11.5 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.

11.5.1 complex.h

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

11.5.2 fenv.h

```
#define FE_INVALID      0x01
#define FE_DIVBYZERO    0x04
#define FE_OVERFLOW     0x08
#define FE_UNDERFLOW    0x10
#define FE_INEXACT      0x20

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

#define FE_TONEAREST    0
#define FE_DOWNWARD     0x400
#define FE_UPWARD       0x800
#define FE_TOWARDZERO   0xc00

typedef unsigned short fexcept_t;

typedef struct {
    unsigned short __control_word;
    unsigned short __unused1;
    unsigned short __status_word;
    unsigned short __unused2;
    unsigned short __tags;
    unsigned short __unused3;
    unsigned int __eip;
    unsigned short __cs_selector;
    unsigned int __opcode:11;
    unsigned int __unused4:5;
    unsigned int __data_offset;
    unsigned short __data_selector;
    unsigned short __unused5;
} fenv_t;

#define FE_DFL_ENV      ((__const fenv_t *) -1)
```

11.5.3 math.h

```
#define fpclassify(x)   \
        (sizeof (x) == sizeof (float) ? __fpclassifyf (x) :sizeof (x)
== sizeof (double) ? __fpclassify (x) : __fpclassifyl (x))
#define signbit(x)      \
```



```
        (sizeof (x) == sizeof (float)? __signbitf (x): sizeof (x) ==
sizeof (double)? __signbit (x) : __signbitl (x))

#define FP_ILOGB0       (-2147483647 - 1)
#define FP_ILOGBNAN     (-2147483647 - 1)

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

11.6 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 11.4 shall behave as described in the referenced base document.

__fpclassifyl

Name

`__fpclassifyl` — test for infinity

Synopsis

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

Description

`__fpclassifyl()` has the same specification as `fpclassifyl()` in ISO POSIX (2003), except that the argument type for `__fpclassifyl()` is known to be long double.

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

11.7 Interfaces for libpthread

Table 11-27 defines the library name and shared object name for the libpthread library

Table 11-27 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] ISO/IEC 23360-1
[SUSv3] ISO POSIX (2003)

11.7.1 Realtime Threads

11.7.1.1 Interfaces for Realtime Threads

An LSB conforming implementation shall provide the architecture specific functions for Realtime Threads specified in Table 11-28, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-28 libpthread - Realtime Threads Function Interfaces

pthread_attr_getinheritsched(GLIBC_2.0) [SUSv3]	pthread_attr_getschedpolicy(GLIBC_2.0) [SUSv3]	pthread_attr_getscope(GLIBC_2.0) [SUSv3]	pthread_attr_setinheritsched(GLIBC_2.0) [SUSv3]
pthread_attr_setschedpolicy(GLIBC_2.0) [SUSv3]	pthread_attr_setscope(GLIBC_2.0) [SUSv3]	pthread_getschedparam(GLIBC_2.0) [SUSv3]	pthread_setschedparam(GLIBC_2.0) [SUSv3]

11.7.2 Advanced Realtime Threads

11.7.2.1 Interfaces for Advanced Realtime Threads

No external functions are defined for libpthread - Advanced Realtime Threads in this part of the specification. See also the generic specification, ISO/IEC 23360-1.

11.7.3 Posix Threads

11.7.3.1 Interfaces for Posix Threads

An LSB conforming implementation shall provide the architecture specific functions for Posix Threads specified in Table 11-29, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-29 libpthread - Posix Threads Function Interfaces

_pthread_cleanup_pop(GLIBC_2.0) [LSB]	_pthread_cleanup_push(GLIBC_2.0) [LSB]	pthread_attr_destroy(GLIBC_2.0) [SUSv3]	pthread_attr_getdetachstate(GLIBC_2.0) [SUSv3]
pthread_attr_getguardsize(GLIBC_2.1) [SUSv3]	pthread_attr_getschedparam(GLIBC_2.0) [SUSv3]	pthread_attr_getstack(GLIBC_2.2) [SUSv3]	pthread_attr_getstackaddr(GLIBC_2.1) [SUSv3]
pthread_attr_getstacksize(GLIBC_2.1) [SUSv3]	pthread_attr_init(GLIBC_2.1) [SUSv3]	pthread_attr_setdetachstate(GLIBC_2.0) [SUSv3]	pthread_attr_setguardsize(GLIBC_2.1) [SUSv3]
pthread_attr_setschedparam(GLIBC_2.0) [SUSv3]	pthread_attr_setstack(GLIBC_2.2) [SUSv3]	pthread_attr_setstackaddr(GLIBC_2.1) [SUSv3]	pthread_attr_setstacksize(GLIBC_2.1) [SUSv3]
pthread_cancel(GLIBC_2.0) [SUSv3]	pthread_cond_broadcast(GLIBC_2.3.2) [SUSv3]	pthread_cond_destroy(GLIBC_2.3.2) [SUSv3]	pthread_cond_init(GLIBC_2.3.2) [SUSv3]
pthread_cond_signal(GLIBC_2.3.2) [SUSv3]	pthread_cond_timedwait(GLIBC_2.3.2) [SUSv3]	pthread_cond_wait(GLIBC_2.3.2) [SUSv3]	pthread_condattr_destroy(GLIBC_2.0) [SUSv3]
pthread_condattr_getpshared(GLIBC_2.2) [SUSv3]	pthread_condattr_init(GLIBC_2.0) [SUSv3]	pthread_condattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_create(GLIBC_2.1) [SUSv3]
pthread_detach(GLIBC_2.0) [SUSv3]	pthread_equal(GLIBC_2.0) [SUSv3]	pthread_exit(GLIBC_2.0) [SUSv3]	pthread_getconcurrency(GLIBC_2.1) [SUSv3]
pthread_getspeci	pthread_join(GLI	pthread_key_cre	pthread_key_del

fic(GLIBC_2.0) [SUSv3]	BC_2.0) [SUSv3]	ate(GLIBC_2.0) [SUSv3]	ete(GLIBC_2.0) [SUSv3]
pthread_kill(GLIBC_2.0) [SUSv3]	pthread_mutex_destroy(GLIBC_2.0) [SUSv3]	pthread_mutex_init(GLIBC_2.0) [SUSv3]	pthread_mutex_lock(GLIBC_2.0) [SUSv3]
pthread_mutex_trylock(GLIBC_2.0) [SUSv3]	pthread_mutex_unlock(GLIBC_2.0) [SUSv3]	pthread_mutexattr_destroy(GLIBC_2.0) [SUSv3]	pthread_mutexattr_getpshared(GLIBC_2.2) [SUSv3]
pthread_mutexattr_gettype(GLIBC_2.1) [SUSv3]	pthread_mutexattr_init(GLIBC_2.0) [SUSv3]	pthread_mutexattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_settype(GLIBC_2.1) [SUSv3]
pthread_once(GLIBC_2.0) [SUSv3]	pthread_rwlock_destroy(GLIBC_2.1) [SUSv3]	pthread_rwlock_init(GLIBC_2.1) [SUSv3]	pthread_rwlock_rdlock(GLIBC_2.1) [SUSv3]
pthread_rwlock_timedrdlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_timedwrlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_tryrdlock(GLIBC_2.1) [SUSv3]	pthread_rwlock_trywrlock(GLIBC_2.1) [SUSv3]
pthread_rwlock_unlock(GLIBC_2.1) [SUSv3]	pthread_rwlock_wrlock(GLIBC_2.1) [SUSv3]	pthread_rwlockattr_destroy(GLIBC_2.1) [SUSv3]	pthread_rwlockattr_getpshared(GLIBC_2.1) [SUSv3]
pthread_rwlockattr_init(GLIBC_2.1) [SUSv3]	pthread_rwlockattr_setpshared(GLIBC_2.1) [SUSv3]	pthread_self(GLIBC_2.0) [SUSv3]	pthread_setcancelstate(GLIBC_2.0) [SUSv3]
pthread_setcanceltype(GLIBC_2.0) [SUSv3]	pthread_setconcurrency(GLIBC_2.1) [SUSv3]	pthread_setspecific(GLIBC_2.0) [SUSv3]	pthread_sigmask(GLIBC_2.0) [SUSv3]
pthread_testcancel(GLIBC_2.0) [SUSv3]	sem_close(GLIBC_2.1.1) [SUSv3]	sem_destroy(GLIBC_2.1) [SUSv3]	sem_getvalue(GLIBC_2.1) [SUSv3]
sem_init(GLIBC_2.1) [SUSv3]	sem_open(GLIBC_2.1.1) [SUSv3]	sem_post(GLIBC_2.1) [SUSv3]	sem_timedwait(GLIBC_2.2) [SUSv3]
sem_trywait(GLIBC_2.1) [SUSv3]	sem_unlink(GLIBC_2.1.1) [SUSv3]	sem_wait(GLIBC_2.1) [SUSv3]	

11.7.4 Thread aware versions of libc interfaces

11.7.4.1 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 11-30, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-30 libpthread - Thread aware versions of libc interfaces Function Interfaces

lseek64(GLIBC_2.2) [LFS]	open64(GLIBC_2.2) [LFS]	pread(GLIBC_2.2) [SUSv3]	pread64(GLIBC_2.2) [LFS]
pwrite(GLIBC_2.2) [SUSv3]	pwrite64(GLIBC_2.2) [LFS]		

11.8 Data Definitions for libpthread

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

11.8.1 pthread.h

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

11.8.2 semaphore.h

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

11.9 Interfaces for libgcc_s

Table 11-31 defines the library name and shared object name for the libgcc_s library

Table 11-31 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] ISO/IEC 23360-1

