Linux Standard Base Core Specification for IA64 3.~~0~~1

Linux Standard Base Core Specification for IA64 3.~~0~~1

Contents

List of Figures

Foreword

This is version 3.~~0~~1 of the Linux Standard Base Core Specification for IA64. This specification is part of a family of specifications under the general title "Linux Standard Base". Developers of applications or implementations interested in using the LSB trademark should see the Free Standards Group Certification Policy for details.

Introduction

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

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

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

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

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

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

1

I Introductory Elements

1 Scope

1.1 General

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

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

The LSB-generic document shall be used in conjunction with an architecture-specific supplement. Whenever a section of the LSB-generic specification shall be supplemented by architecture-specific information, the LSB-generic document includes a reference to the architecture supplement. Architecture supplements 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 shall provide 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 Itanium architecture specific Core module of the Linux Standards Base (LSB). This module supplements the generic LSB Core module with those interfaces that differ between architectures.

Interfaces described in this module are mandatory except where explicitly listed otherwise. Core interfaces may be supplemented by other modules; all modules are built upon the core.

~~2 Normative References~~

~~The specifications listed below are referenced in whole or in part by the Linux Standard Base. In this specification, where only a particular section of one of these references is identified, then the normative reference is to that section alone, and the rest of the referenced document is informative.~~

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
~~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/~~
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEC 60559/IEEE 754 Floating Point	IEC 60559:1989 Binary floating-point arithmetic for microprocessor systems	http://www.ieee.org/
Intel® Itanium ™ Processor-specific Application Binary Interface	Intel® Itanium ™ Processor-specific Application Binary Interface	http://refspecs.freestandards.org/elf/IA64-SysV-psABI.pdf
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)	http://www.unix.org/version3/

Name	Title	URL
	-- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004	
~~ISO/IEC TR14652~~	~~ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions~~	
Itanium ™ Architecture Software Developer's Manual Volume 1	Itanium ™ Architecture Software Developer's Manual Volume 1: Application Architecture	http://refspecs.freestandards.org/IA64-softdevman-vol1.pdf
Itanium ™ Architecture Software Developer's Manual Volume 2	Itanium ™ Architecture Software Developer's Manual Volume 2: System Architecture	http://refspecs.freestandards.org/IA64-softdevman-vol2.pdf
Itanium ™ Architecture Software Developer's Manual Volume 3	Itanium ™ Architecture Software Developer's Manual Volume 3: Instruction Set Reference	http://refspecs.freestandards.org/IA64-softdevman-vol3.pdf
Itanium ™ Architecture Software Developer's Manual Volume 4	IA-64 Processor Reference: Intel® Itanium ™ Processor Reference Manual for Software Development	http://refspecs.freestandards.org/IA64-softdevman-vol4.pdf
Itanium ™ Software Conventions and Runtime Guide	Itanium ™ Software Conventions & Runtime Architecture Guide, September 2000	http://refspecs.freestandards.org/IA64conventions.pdf

Name	Title	URL
~~ITU-T V.42~~	~~International Telecommunication Union Recommendation V.42 (2002): Error-correcting procedures for DCEs using asynchronous-to-synchronous conversionITUV~~	~~http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42~~
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
~~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~~
~~PAM~~	~~Open Software Foundation, Request For Comments: 86.0 , October 1995, V. Samar & R.Schemers (SunSoft)~~	~~http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt~~
~~RFC 1321: The MD5 Message-Digest Algorithm~~	~~IETF RFC 1321: The MD5 Message-Digest Algorithm~~	~~http://www.ietf.org/rfc/rfc1321.txt~~
~~RFC 1833: Binding Protocols for ONC RPC Version 2~~	~~IETF RFC 1833: Binding Protocols for ONC RPC Version 2~~	~~http://www.ietf.org/rfc/rfc1833.txt~~
~~RFC 1950: ZLIB Compressed Data Format Specication~~	~~IETF RFC 1950: ZLIB Compressed Data Format Specification~~	~~http://www.ietf.org/rfc/rfc1950.txt~~
~~RFC 1951: DEFLATE Compressed Data Format Specification~~	~~IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3~~	~~http://www.ietf.org/rfc/rfc1951.txt~~
~~RFC 1952: GZIP File Format Specification~~	~~IETF RFC 1952: GZIP file format specification version 4.3~~	~~http://www.ietf.org/rfc/rfc1952.txt~~
~~RFC 2440: OpenPGP Message Format~~	~~IETF RFC 2440: OpenPGP Message Format~~	~~http://www.ietf.org/rfc/rfc2440.txt~~
~~RFC 2821:Simple Mail~~	~~IETF RFC 2821: Simple~~	~~http://www.ietf.org/rfc~~

Name	Title	URL
~~Transfer Protocol~~	~~Mail Transfer Protocol~~	~~/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~~
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
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
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
~~this specification~~	~~Linux Standard Base~~	~~http://www.linuxbase.org/spec/~~
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/
ISO/IEC TR14652	ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions	
ITU-T V.42	International Telecommunication Union Recommendation V.42 (2002): Error-correcting procedures for DCEs using asynchronous-to-synchronous conversionITUV	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Li18nux Globalization Specification	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt
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

Name	Title	URL
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
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 IA64 Linux Standard Base systems, with the specified runtime names. These names override or supplement the names specified in the generic LSB 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.6.1
libdl	libdl.so.2
libcrypt	libcrypt.so.1
libz	libz.so.1
libncurses	libncurses.so.5
libutil	libutil.so.1
libc	libc.so.6.1
libpthread	libpthread.so.0
proginterp	/lib/ld-lsb-ia64.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 and its relevant architecture specific supplement.

> **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:

- ~~The implementation shall implement fully the architecture described in the hardware manual for the target processor architecture.~~
- A processor architecture represents a family of related processors which may not have identical feature sets. The architecture specific supplement to this specification for a given target processor architecture describes 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 and its relevant architecture specific supplement.

A conforming application shall satisfy the following requirements:

- Its executable files ~~are~~ 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 ~~employs~~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 ~~is~~shall be stated in the application's documentation.
- It ~~does~~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 ~~is~~shall be in turn an LSB conforming application.
    - The use of that interface or data format, as well as its source, ~~is~~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 ~~does~~shall not require or use any interface, facility, or implementation-defined extension that is not defined in this document in order to be installed or to execute successfully.

4 Definitions

For the purposes of this document, the following definitions, as specified in the *ISO/IEC Directives, Part 2, 2001, 4th Edition*, apply:

can

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

cannot

> be unable to; there is no 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) [~~1]~~SUSv3]

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

> **Note:** Symbol versions are defined in the ~~listed references below the table.~~architecture specific supplements 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 Intel® Itanium ™ Processor-specific Application Binary Interface, 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 Itanium™ Architecture is specified by the following documents

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

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

~~Only~~Conforming applications may use only instructions which do not require elevated privileges~~ may be used by~~.

Conforming applications shall not invoke the ~~application.~~

~~Applications may not make~~implementations underlying system ~~calls~~call interface directly. The interfaces in the implementation base libraries ~~must~~ shall be used instead.

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

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

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

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

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

8.1.2 Data Representation

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

Within this specification, the term `byte` refers to an 8-bit object, the term `halfword` refers to a 16-bit object, the term `word` refers to a 32-bit object, the term `doubleword` refers to a 64-bit object, and the term `quadword` refers to a 128-bit object.

8.1.2.1 Byte Ordering

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

8.1.2.2 Fundamental Types

Table 8-1 describes how fundemental C language data types shall be represented:

Table 8-1 Scalar Types

Type	**C**	**`sizeof`**	**Alignment (bytes)**	**Hardware Representation**
Integral	_Bool	1	1	byte (sign unspecified)
	char	1	1	signed byte
	signed char			
	unsigned char			signed byte
	short	2	2	signed half-word
	signed short			
	unsigned short			unsigned halfword
	int	4	4	signed word
	signed int			
	unsigned int			unsigned word
	long	8	8	signed doubleword
	signed long			
	unsigned long			unsigned doubleword
	long long	8	8	signed doubleword
	signed long long			

Type	C	`sizeof`	Alignment (bytes)	Hardware Representation
	unsigned long long			unsigned doubleword
Pointer	*`any-type`* *	8	8	unsigned doubleword
	`any-type (*)()`			
Floating-Point	float	4	4	IEEE Single-precision
	double	8	8	IEEE Double-precision
	long double	16	16	IEEE Double-extended

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

8.1.2.3 Aggregates and Unions

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

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

A conforming application shall not read padding.

```
struct {
    char c;
}
```

Byte aligned, `sizeof` is 1

Offset	Byte 0
0	c^0

Figure 8-1 Structure Smaller Than A Word

```
struct {
    char  c;
    char  d;
    short s;
    int   i;
```

```
        long  l;
    }
```

Doubleword Aligned, `sizeof` is 16

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

Figure 8-2 No Padding

```
    struct {
        char  c;
        long  l;
        int   i;
        short s;
    }
```

Doubleword Aligned, `sizeof` is 24

Offset	Byte 3	Byte 2	Byte 1	Byte 0
0	pad^1			c^0
4	pad^1			
8	l^0			
12				
16	i^0			
20	pad^2		s^0	

Figure 8-3 Internal and Tail Padding

8.1.2.4 Bit Fields

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

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

- Bit-fields are allocated from right to left (least to most significant).
- A bit-field must entirely reside in a storage unit for its appropriate type. A bit field shall never cross its unit boundary.
- Bit-fields may share a storage unit with other `struct/union` members, including members that are not bit fields. Such other `struct/union` members shall occupy different parts of the storage unit.

- The type of unnamed bit-fields shall not affect the alignment of a structure or union, although individual bit-field member offsets shall obey the alignment constraints.

Bit-field Type	Width *w*	Range

`signed char`
`char`
`unsigned char`
	1 to 8	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

`signed short`
`short`
`unsigned short`
	1 to 16	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

`signed int`
`int`
`unsigned int`
	1 to 32	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

`signed long`
`long`
`unsigned long`
	1 to 64	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

Figure 8-4 Bit-Field Ranges

8.2 Function Calling Sequence

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

8.2.1 Registers

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

8.2.2 Floating Point Registers

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

8.2.3 Stack Frame

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

8.2.4 Arguments

8.2.4.1 Introduction

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

8.2.4.2 Integral/Pointer

See Itanium ™ Software Conventions and Runtime Guide Chapter 8.5.

8.2.4.3 Floating Point

See Itanium ™ Software Conventions and Runtime Guide Chapter 8.5.

8.2.4.4 Struct and Union Point

See Itanium ™ Software Conventions and Runtime Guide Chapter 8.5.

8.2.4.5 Variable Arguments

See Itanium ™ Software Conventions and Runtime Guide Chapter 8.5.4.

8.2.5 Return Values

8.2.5.1 Introduction

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

8.2.5.2 Void

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

8.2.5.3 Integral/Pointer

See Itanium ™ Software Conventions and Runtime Guide Chapter 8.6.

8.2.5.4 Floating Point

See Itanium ™ Software Conventions and Runtime Guide Chapter 8.6.

8.2.5.5 Struct and Union

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

8.3 Operating System Interface

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

8.3.1 Processor Execution Mode

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

8.3.2 Exception Interface

8.3.2.1 Introduction

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

8.3.2.2 Hardware Exception Types

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

8.3.2.3 Software Trap Types

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

8.3.3 Signal Delivery

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

8.3.3.1 Signal Handler Interface

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

8.3.4 Debugging Support

The LSB does not specify debugging information.

8.3.5 Process Startup

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

8.4 Process Initialization

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

8.4.1 Special Registers

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

8.4.2 Process Stack (on entry)

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

8.4.3 Auxiliary Vector

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

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

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

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

AT_NULL

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

AT_IGNORE

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

AT_EXECFD

: File descriptor of program

AT_PHDR

: Program headers for program

AT_PHENT

: Size of program header entry

AT_PHNUM

: Number of program headers

AT_PAGESZ

: System page size

AT_BASE

: Base address of interpreter

AT_FLAGS

: Flags

AT_ENTRY

: Entry point of program

AT_NOTELF

: Program is not ELF

AT_UID

: Real uid

AT_EUID

: Effective uid

AT_GID

: Real gid

AT_EGID

: Effective gid

AT_CLKTCK

: Frequency of times()

AT_PLATFORM

String identifying platform.

AT_HWCAP

Machine dependent hints about processor capabilities.

AT_FPUCW

Used FPU control word

AT_DCACHEBSIZE

Data cache block size

AT_ICACHEBSIZE

Instruction cache block size

AT_UCACHEBSIZE

Unified cache block size

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

8.4.4 Environment

Although a pointer to the environment vector should be available as a third argument to the `main()` entry point, conforming applications should use `getenv()` to access the environment. (See ISO POSIX (2003), Section `exec()`).

8.5 Coding Examples

8.5.1 Introduction

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

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

8.5.2 Code Model Overview/Architecture Constraints

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

8.5.3 Position-Independent Function Prologue

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

8.5.4 Data Objects

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

8.5.4.1 Absolute Load & Store

Conforming applications shall not use absolute addressing.

8.5.4.2 Position Relative Load & Store

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

8.5.5 Function Calls

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

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

8.5.5.1 Absolute Direct Function Call

Conforming applications shall not use absolute addressing.

8.5.5.2 Absolute Indirect Function Call

Conforming applications shall not use absolute addressing.

8.5.5.3 Position-Independent Direct Function Call

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

8.5.5.4 Position-Independent Indirect Function Call

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

8.5.6 Branching

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

8.5.6.1 Branch Instruction

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

8.5.6.2 Absolute switch() code

Conforming applications shall not use absolute addressing.

8.5.6.3 Position-Independent switch() code

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

8.6 C Stack Frame

8.6.1 Variable Argument List

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

8.6.2 Dynamic Allocation of Stack Space

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

8.7 Debug Information

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

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, Intel® Itanium ™ Processor-specific Application Binary Interface and as supplemented by the Linux Standard Base Specification and this document.

9.2 ELF Header

9.2.1 Machine Information

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

9.2.1.1 File Class

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

9.2.1.2 Data Encoding

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

9.2.1.3 OS Identification

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

9.2.1.4 Processor Identification

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

9.2.1.5 Processor Specific Flags

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

The following additional processor-specific flags are defined:

Table 9-1 Additional Processor-Specific Flags

Name	Value
EF_IA_64_LINUX_EXECUTABLE_STACK	0x00000001

EF_IA_64_LINUX_EXECUTABLE_STACK

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

9.3 Sections

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

9.3.1 Special Sections

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

Table 9-2 ELF Special Sections

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

.got

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

.IA_64.archext

This section holds product-specific extension bits. The link editor will perform a logical "or" of the extension bits of each object when creating an executable so that it creates only a single .IA_64.archext section in the executable.

.IA_64.pltoff

This section holds local function descriptor entries.

.IA_64.unwind

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

.IA_64.unwind_info

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

.plt

This section holds the Procedure Linkage Table.

.sbss

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

.sdata

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

.sdata1

See .sdata.

9.3.2 Linux Special Sections

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

Table 9-3 Additional Special Sections

Name	Type	Attributes
.opd	SHT_PROGBITS	SHF_ALLOC
.rela.dyn	SHT_RELA	SHF_ALLOC
.rela.IA_64.pltoff	SHT_RELA	SHF_ALLOC

.opd

This section holds function descriptors

.rela.dyn

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

.rela.IA_64.pltoff

This section holds relocation information, as described in `Relocation'. These relocations are applied to the .IA_64.pltoff section.

9.3.3 Section Types

Section Types are described in the Intel® Itanium ™ Processor-specific Application Binary Interface, Chapter 4.2. LSB conforming implementations are not required to use any sections in the range from `SHT_IA_64_LOPSREG` to `SHT_IA_64_HIPSREG`. Additionally, LSB conforming implementations are not required to support the `SHT_IA_64_PRIORITY_INIT` section, beyond the gABI requirements for the handling of unrecognized section types, linking them into a contiguous section in the object file created by the static linker.

9.3.4 Section Attribute Flags

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

9.3.5 Special Section Types

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

9.4 Symbol Table

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

9.5 Relocation

9.5.1 Relocation Types

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

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, Intel® Itanium ™ Processor-specific Application Binary Interface and as supplemented by the Linux Standard Base Specification and this document.

10.2 Program Header

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

10.2.1 Types

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

10.2.2 Flags

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

10.3 Program Loading

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

10.4 Dynamic Linking

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

10.4.1 Dynamic Entries

10.4.1.1 ELF Dynamic Entries

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

DT_PLTGOT

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

10.4.1.2 Additional Dynamic Entries

The following dynamic entries are defined here.

DT_RELACOUNT

The number of relative relocations in .rela.dyn

10.4.2 Global Offset Table

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

10.4.3 Shared Object Dependencies

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

10.4.4 Function Addresses

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

10.4.5 Procedure Linkage Table

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

10.4.6 Initialization and Termination Functions

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

III Base Libraries

11 Libraries

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

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

11.1 Program Interpreter/Dynamic Linker

The ~~LSB specifies the~~ Program Interpreter ~~to~~shall be `/lib/ld-lsb-ia64.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.6.1

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

[LFS] Large File Support
[LSB] ~~this specification~~This Specification
[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_2.2) [1]~~	~~svc_getreqset(GLIBC_2.2) [2]~~	~~svcudp_create(GLIBC_2.2) [3]~~	~~xdr_int(GLIBC_2.2) [2]~~	~~xdr_u_long(GLIBC_2.2) [2]~~
~~clnt_create(GLIBC_2.2) [1]~~	~~svc_register(GLIBC_2.2) [3]~~	~~xdr_accepted_reply(GLIBC_2.2) [2]~~	~~xdr_long(GLIBC_2.2) [2]~~	~~xdr_u_short(GLIBC_2.2) [2]~~
~~clnt_pcreateerror(GLIBC_2.2) [1]~~	~~svc_run(GLIBC_2.2) [3]~~	~~xdr_array(GLIBC_2.2) [2]~~	~~xdr_opaque(GLIBC_2.2) [2]~~	~~xdr_union(GLIBC_2.2) [2]~~
~~clnt_perrno(G~~	~~svc_sendrepl~~	~~xdr_bool(GLI~~	~~xdr_opaque_a~~	~~xdr_vector(G~~

~~LIBC_2.2) [1]~~	~~y(GLIBC_2.2) [3]~~	~~BC_2.2) [2]~~	~~uth(GLIBC_2.2) [2]~~	~~LIBC_2.2) [2]~~
~~clnt_perror(GLIBC_2.2) [1]~~	~~svcerr_auth(GLIBC_2.2) [2]~~	~~xdr_bytes(GLIBC_2.2) [2]~~	~~xdr_pointer(GLIBC_2.2) [2]~~	~~xdr_void(GLIBC_2.2) [2]~~
~~clnt_spcreateerror(GLIBC_2.2) [1]~~	~~svcerr_decode(GLIBC_2.2) [2]~~	~~xdr_callhdr(GLIBC_2.2) [2]~~	~~xdr_reference(GLIBC_2.2) [2]~~	~~xdr_wrapstring(GLIBC_2.2) [2]~~
~~clnt_sperrno(GLIBC_2.2) [1]~~	~~svcerr_noproc(GLIBC_2.2) [2]~~	~~xdr_callmsg(GLIBC_2.2) [2]~~	~~xdr_rejected_reply(GLIBC_2.2) [2]~~	~~xdrmem_create(GLIBC_2.2) [2]~~
~~clnt_sperror(GLIBC_2.2) [1]~~	~~svcerr_noprog(GLIBC_2.2) [2]~~	~~xdr_char(GLIBC_2.2) [2]~~	~~xdr_replymsg(GLIBC_2.2) [2]~~	~~xdrrec_create(GLIBC_2.2) [2]~~
~~key_decryptsession(GLIBC_2.2) [2]~~	~~svcerr_progvers(GLIBC_2.2) [2]~~	~~xdr_double(GLIBC_2.2) [2]~~	~~xdr_short(GLIBC_2.2) [2]~~	~~xdrrec_eof(GLIBC_2.2) [2]~~
~~pmap_getport(GLIBC_2.2) [3]~~	~~svcerr_systemerr(GLIBC_2.2) [2]~~	~~xdr_enum(GLIBC_2.2) [2]~~	~~xdr_string(GLIBC_2.2) [2]~~	
~~pmap_set(GLIBC_2.2) [3]~~	~~svcerr_weakauth(GLIBC_2.2) [2]~~	~~xdr_float(GLIBC_2.2) [2]~~	~~xdr_u_char(GLIBC_2.2) [2]~~	
~~pmap_unset(GLIBC_2.2) [3]~~	~~svctcp_create(GLIBC_2.2) [3]~~	~~xdr_free(GLIBC_2.2) [2]~~	~~xdr_u_int(GLIBC_2.2) [3]~~	

~~*Referenced Specification(s)*~~

~~**[1].** SVID Issue 4~~

~~**[2].** SVID Issue 3~~

~~**[3].** this specification~~

authnone_create(GLIBC_2.2) [SVID.4]	clnt_create(GLIBC_2.2) [SVID.4]	clnt_pcreateerror(GLIBC_2.2) [SVID.4]	clnt_perrno(GLIBC_2.2) [SVID.4]
clnt_perror(GLIBC_2.2) [SVID.4]	clnt_spcreateerror(GLIBC_2.2) [SVID.4]	clnt_sperrno(GLIBC_2.2) [SVID.4]	clnt_sperror(GLIBC_2.2) [SVID.4]
key_decryptsession(GLIBC_2.2) [SVID.3]	pmap_getport(GLIBC_2.2) [LSB]	pmap_set(GLIBC_2.2) [LSB]	pmap_unset(GLIBC_2.2) [LSB]
svc_getreqset(GLIBC_2.2) [SVID.3]	svc_register(GLIBC_2.2) [LSB]	svc_run(GLIBC_2.2) [LSB]	svc_sendreply(GLIBC_2.2) [LSB]
svcerr_auth(GLIBC_2.2) [SVID.3]	svcerr_decode(GLIBC_2.2) [SVID.3]	svcerr_noproc(GLIBC_2.2) [SVID.3]	svcerr_noprog(GLIBC_2.2) [SVID.3]
svcerr_progvers(	svcerr_systemerr(	svcerr_weakauth(	svctcp_create(GLI

GLIBC_2.2) [SVID.3]	GLIBC_2.2) [SVID.3]	GLIBC_2.2) [SVID.3]	BC_2.2) [LSB]
svcudp_create(GLIBC_2.2) [LSB]	xdr_accepted_reply(GLIBC_2.2) [SVID.3]	xdr_array(GLIBC_2.2) [SVID.3]	xdr_bool(GLIBC_2.2) [SVID.3]
xdr_bytes(GLIBC_2.2) [SVID.3]	xdr_callhdr(GLIBC_2.2) [SVID.3]	xdr_callmsg(GLIBC_2.2) [SVID.3]	xdr_char(GLIBC_2.2) [SVID.3]
xdr_double(GLIBC_2.2) [SVID.3]	xdr_enum(GLIBC_2.2) [SVID.3]	xdr_float(GLIBC_2.2) [SVID.3]	xdr_free(GLIBC_2.2) [SVID.3]
xdr_int(GLIBC_2.2) [SVID.3]	xdr_long(GLIBC_2.2) [SVID.3]	xdr_opaque(GLIBC_2.2) [SVID.3]	xdr_opaque_auth(GLIBC_2.2) [SVID.3]
xdr_pointer(GLIBC_2.2) [SVID.3]	xdr_reference(GLIBC_2.2) [SVID.3]	xdr_rejected_reply(GLIBC_2.2) [SVID.3]	xdr_replymsg(GLIBC_2.2) [SVID.3]
xdr_short(GLIBC_2.2) [SVID.3]	xdr_string(GLIBC_2.2) [SVID.3]	xdr_u_char(GLIBC_2.2) [SVID.3]	xdr_u_int(GLIBC_2.2) [LSB]
xdr_u_long(GLIBC_2.2) [SVID.3]	xdr_u_short(GLIBC_2.2) [SVID.3]	xdr_union(GLIBC_2.2) [SVID.3]	xdr_vector(GLIBC_2.2) [SVID.3]
xdr_void(GLIBC_2.2) [SVID.3]	xdr_wrapstring(GLIBC_2.2) [SVID.3]	xdrmem_create(GLIBC_2.2) [SVID.3]	xdrrec_create(GLIBC_2.2) [SVID.3]
xdrrec_eof(GLIBC_2.2) [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_2.2) [1]~~	~~fchmod(GLIBC_2.2) [2]~~	~~getwd(GLIBC_2.2) [2]~~	~~read(GLIBC_2.2) [2]~~	~~setrlimit(GLIBC_2.2) [2]~~
~~__getpgid(GLIBC_2.2) [1]~~	~~fchown(GLIBC_2.2) [2]~~	~~initgroups(GLIBC_2.2) [1]~~	~~readdir(GLIBC_2.2) [2]~~	~~setrlimit64(GLIBC_2.2) [3]~~
~~__lxstat(GLIBC_2.2) [1]~~	~~fcntl(GLIBC_2.2) [1]~~	~~ioctl(GLIBC_2.2) [1]~~	~~readdir_r(GLIBC_2.2) [2]~~	~~setsid(GLIBC_2.2) [2]~~
~~__xmknod(GLIBC_2.2) [1]~~	~~fdatasync(GLIBC_2.2) [2]~~	~~kill(GLIBC_2.2) [1]~~	~~readlink(GLIBC_2.2) [2]~~	~~setuid(GLIBC_2.2) [2]~~
~~__xstat(GLIBC_2.2) [1]~~	~~flock(GLIBC_2.2) [1]~~	~~killpg(GLIBC_2.2) [2]~~	~~readv(GLIBC_2.2) [2]~~	~~sleep(GLIBC_2.2) [2]~~
~~access(GLIBC~~	~~fork(GLIBC_2~~	~~lchown(GLIB~~	~~rename(GLIB~~	~~statvfs(GLIBC~~

~~_2.2) [2]~~	~~.2) [2]~~	~~C_2.2) [2]~~	~~C_2.2) [2]~~	~~_2.2) [2]~~
~~acct(GLIBC_2.2) [1]~~	~~fstatvfs(GLIBC_2.2) [2]~~	~~link(GLIBC_2.2) [1]~~	~~rmdir(GLIBC_2.2) [2]~~	~~stime(GLIBC_2.2) [1]~~
~~alarm(GLIBC_2.2) [2]~~	~~fsync(GLIBC_2.2) [2]~~	~~lockf(GLIBC_2.2) [2]~~	~~sbrk(GLIBC_2.2) [4]~~	~~symlink(GLIBC_2.2) [2]~~
~~brk(GLIBC_2.2) [4]~~	~~ftime(GLIBC_2.2) [2]~~	~~lseek(GLIBC_2.2) [2]~~	~~sched_get_priority_max(GLIBC_2.2) [2]~~	~~sync(GLIBC_2.2) [2]~~
~~chdir(GLIBC_2.2) [2]~~	~~ftruncate(GLIBC_2.2) [2]~~	~~mkdir(GLIBC_2.2) [2]~~	~~sched_get_priority_min(GLIBC_2.2) [2]~~	~~sysconf(GLIBC_2.2) [2]~~
~~chmod(GLIBC_2.2) [2]~~	~~getcontext(GLIBC_2.2) [2]~~	~~mkfifo(GLIBC_2.2) [2]~~	~~sched_getparam(GLIBC_2.2) [2]~~	~~time(GLIBC_2.2) [2]~~
~~chown(GLIBC_2.2) [2]~~	~~getegid(GLIBC_2.2) [2]~~	~~mlock(GLIBC_2.2) [2]~~	~~sched_getscheduler(GLIBC_2.2) [2]~~	~~times(GLIBC_2.2) [2]~~
~~chroot(GLIBC_2.2) [4]~~	~~geteuid(GLIBC_2.2) [2]~~	~~mlockall(GLIBC_2.2) [2]~~	~~sched_rr_get_interval(GLIBC_2.2) [2]~~	~~truncate(GLIBC_2.2) [2]~~
~~clock(GLIBC_2.2) [2]~~	~~getgid(GLIBC_2.2) [2]~~	~~mmap(GLIBC_2.2) [2]~~	~~sched_setparam(GLIBC_2.2) [2]~~	~~ulimit(GLIBC_2.2) [2]~~
~~close(GLIBC_2.2) [2]~~	~~getgroups(GLIBC_2.2) [2]~~	~~mprotect(GLIBC_2.2) [2]~~	~~sched_setscheduler(GLIBC_2.2) [2]~~	~~umask(GLIBC_2.2) [2]~~
~~closedir(GLIBC_2.2) [2]~~	~~getitimer(GLIBC_2.2) [2]~~	~~msync(GLIBC_2.2) [2]~~	~~sched_yield(GLIBC_2.2) [2]~~	~~uname(GLIBC_2.2) [2]~~
~~creat(GLIBC_2.2) [2]~~	~~getloadavg(GLIBC_2.2) [1]~~	~~munlock(GLIBC_2.2) [2]~~	~~select(GLIBC_2.2) [2]~~	~~unlink(GLIBC_2.2) [1]~~
~~dup(GLIBC_2.2) [2]~~	~~getpagesize(GLIBC_2.2) [4]~~	~~munlockall(GLIBC_2.2) [2]~~	~~setcontext(GLIBC_2.2) [2]~~	~~utime(GLIBC_2.2) [2]~~
~~dup2(GLIBC_2.2) [2]~~	~~getpgid(GLIBC_2.2) [2]~~	~~munmap(GLIBC_2.2) [2]~~	~~setegid(GLIBC_2.2) [2]~~	~~utimes(GLIBC_2.2) [2]~~
~~execl(GLIBC_2.2) [2]~~	~~getpgrp(GLIBC_2.2) [2]~~	~~nanosleep(GLIBC_2.2) [2]~~	~~seteuid(GLIBC_2.2) [2]~~	~~vfork(GLIBC_2.2) [2]~~
~~execle(GLIBC_2.2) [2]~~	~~getpid(GLIBC_2.2) [2]~~	~~nice(GLIBC_2.2) [2]~~	~~setgid(GLIBC_2.2) [2]~~	~~wait(GLIBC_2.2) [2]~~
~~execlp(GLIBC_2.2) [2]~~	~~getppid(GLIBC_2.2) [2]~~	~~open(GLIBC_2.2) [2]~~	~~setitimer(GLIBC_2.2) [2]~~	~~wait4(GLIBC_2.2) [1]~~
~~execv(GLIBC_2.2) [2]~~	~~getpriority(GLIBC_2.2) [2]~~	~~opendir(GLIBC_2.2) [2]~~	~~setpgid(GLIBC_2.2) [2]~~	~~waitpid(GLIBC_2.2) [1]~~

~~execve(GLIBC_2.2) [2]~~	~~getrlimit(GLIBC_2.2) [2]~~	~~pathconf(GLIBC_2.2) [2]~~	~~setpgrp(GLIBC_2.2) [2]~~	~~write(GLIBC_2.2) [2]~~
~~execvp(GLIBC_2.2) [2]~~	~~getrusage(GLIBC_2.2) [2]~~	~~pause(GLIBC_2.2) [2]~~	~~setpriority(GLIBC_2.2) [2]~~	~~writev(GLIBC_2.2) [2]~~
~~exit(GLIBC_2.2) [2]~~	~~getsid(GLIBC_2.2) [2]~~	~~pipe(GLIBC_2.2) [2]~~	~~setregid(GLIBC_2.2) [2]~~	
~~fchdir(GLIBC_2.2) [2]~~	~~getuid(GLIBC_2.2) [2]~~	~~poll(GLIBC_2.2) [2]~~	~~setreuid(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

~~**[2].** ISO POSIX (2003)~~

~~**[3].** Large File Support~~

~~**[4].** SUSv2~~

__fxstat(GLIBC_2.2) [LSB]	__getpgid(GLIBC_2.2) [LSB]	__lxstat(GLIBC_2.2) [LSB]	__xmknod(GLIBC_2.2) [LSB]
__xstat(GLIBC_2.2) [LSB]	access(GLIBC_2.2) [SUSv3]	acct(GLIBC_2.2) [LSB]	alarm(GLIBC_2.2) [SUSv3]
brk(GLIBC_2.2) [SUSv2]	chdir(GLIBC_2.2) [SUSv3]	chmod(GLIBC_2.2) [SUSv3]	chown(GLIBC_2.2) [SUSv3]
chroot(GLIBC_2.2) [SUSv2]	clock(GLIBC_2.2) [SUSv3]	close(GLIBC_2.2) [SUSv3]	closedir(GLIBC_2.2) [SUSv3]
creat(GLIBC_2.2) [SUSv3]	dup(GLIBC_2.2) [SUSv3]	dup2(GLIBC_2.2) [SUSv3]	execl(GLIBC_2.2) [SUSv3]
execle(GLIBC_2.2) [SUSv3]	execlp(GLIBC_2.2) [SUSv3]	execv(GLIBC_2.2) [SUSv3]	execve(GLIBC_2.2) [SUSv3]
execvp(GLIBC_2.2) [SUSv3]	exit(GLIBC_2.2) [SUSv3]	fchdir(GLIBC_2.2) [SUSv3]	fchmod(GLIBC_2.2) [SUSv3]
fchown(GLIBC_2.2) [SUSv3]	fcntl(GLIBC_2.2) [LSB]	fdatasync(GLIBC_2.2) [SUSv3]	flock(GLIBC_2.2) [LSB]
fork(GLIBC_2.2) [SUSv3]	fstatvfs(GLIBC_2.2) [SUSv3]	fsync(GLIBC_2.2) [SUSv3]	ftime(GLIBC_2.2) [SUSv3]
ftruncate(GLIBC_2.2) [SUSv3]	getcontext(GLIBC_2.2) [SUSv3]	getegid(GLIBC_2.2) [SUSv3]	geteuid(GLIBC_2.2) [SUSv3]
getgid(GLIBC_2.2) [SUSv3]	getgroups(GLIBC_2.2) [SUSv3]	getitimer(GLIBC_2.2) [SUSv3]	getloadavg(GLIBC_2.2) [LSB]
getpagesize(GLIBC_2.2) [SUSv2]	getpgid(GLIBC_2.2) [SUSv3]	getpgrp(GLIBC_2.2) [SUSv3]	getpid(GLIBC_2.2) [SUSv3]
getppid(GLIBC_2.2) [SUSv3]	getpriority(GLIBC_2.2) [SUSv3]	getrlimit(GLIBC_2.2) [SUSv3]	getrusage(GLIBC_2.2) [SUSv3]
getsid(GLIBC_2.2)	getuid(GLIBC_2.2	getwd(GLIBC_2.2	initgroups(GLIBC

[SUSv3]	) [SUSv3]	) [SUSv3]	_2.2) [LSB]
ioctl(GLIBC_2.2) [LSB]	kill(GLIBC_2.2) [LSB]	killpg(GLIBC_2.2) [SUSv3]	lchown(GLIBC_2.2) [SUSv3]
link(GLIBC_2.2) [LSB]	lockf(GLIBC_2.2) [SUSv3]	lseek(GLIBC_2.2) [SUSv3]	mkdir(GLIBC_2.2) [SUSv3]
mkfifo(GLIBC_2.2) [SUSv3]	mlock(GLIBC_2.2) [SUSv3]	mlockall(GLIBC_2.2) [SUSv3]	mmap(GLIBC_2.2) [SUSv3]
mprotect(GLIBC_2.2) [SUSv3]	msync(GLIBC_2.2) [SUSv3]	munlock(GLIBC_2.2) [SUSv3]	munlockall(GLIBC_2.2) [SUSv3]
munmap(GLIBC_2.2) [SUSv3]	nanosleep(GLIBC_2.2) [SUSv3]	nice(GLIBC_2.2) [SUSv3]	open(GLIBC_2.2) [SUSv3]
opendir(GLIBC_2.2) [SUSv3]	pathconf(GLIBC_2.2) [SUSv3]	pause(GLIBC_2.2) [SUSv3]	pipe(GLIBC_2.2) [SUSv3]
poll(GLIBC_2.2) [SUSv3]	read(GLIBC_2.2) [SUSv3]	readdir(GLIBC_2.2) [SUSv3]	readdir_r(GLIBC_2.2) [SUSv3]
readlink(GLIBC_2.2) [SUSv3]	readv(GLIBC_2.2) [SUSv3]	rename(GLIBC_2.2) [SUSv3]	rmdir(GLIBC_2.2) [SUSv3]
sbrk(GLIBC_2.2) [SUSv2]	sched_get_priority_max(GLIBC_2.2) [SUSv3]	sched_get_priority_min(GLIBC_2.2) [SUSv3]	sched_getparam(GLIBC_2.2) [SUSv3]
sched_getscheduler(GLIBC_2.2) [SUSv3]	sched_rr_get_interval(GLIBC_2.2) [SUSv3]	sched_setparam(GLIBC_2.2) [SUSv3]	sched_setscheduler(GLIBC_2.2) [SUSv3]
sched_yield(GLIBC_2.2) [SUSv3]	select(GLIBC_2.2) [SUSv3]	setcontext(GLIBC_2.2) [SUSv3]	setegid(GLIBC_2.2) [SUSv3]
seteuid(GLIBC_2.2) [SUSv3]	setgid(GLIBC_2.2) [SUSv3]	setitimer(GLIBC_2.2) [SUSv3]	setpgid(GLIBC_2.2) [SUSv3]
setpgrp(GLIBC_2.2) [SUSv3]	setpriority(GLIBC_2.2) [SUSv3]	setregid(GLIBC_2.2) [SUSv3]	setreuid(GLIBC_2.2) [SUSv3]
setrlimit(GLIBC_2.2) [SUSv3]	setrlimit64(GLIBC_2.2) [LFS]	setsid(GLIBC_2.2) [SUSv3]	setuid(GLIBC_2.2) [SUSv3]
sleep(GLIBC_2.2) [SUSv3]	statvfs(GLIBC_2.2) [SUSv3]	stime(GLIBC_2.2) [LSB]	symlink(GLIBC_2.2) [SUSv3]
sync(GLIBC_2.2) [SUSv3]	sysconf(GLIBC_2.2) [SUSv3]	time(GLIBC_2.2) [SUSv3]	times(GLIBC_2.2) [SUSv3]
truncate(GLIBC_2.2) [SUSv3]	ulimit(GLIBC_2.2) [SUSv3]	umask(GLIBC_2.2) [SUSv3]	uname(GLIBC_2.2) [SUSv3]
unlink(GLIBC_2.2) [LSB]	utime(GLIBC_2.2) [SUSv3]	utimes(GLIBC_2.2) [SUSv3]	vfork(GLIBC_2.2) [SUSv3]
wait(GLIBC_2.2) [SUSv3]	wait4(GLIBC_2.2) [LSB]	waitpid(GLIBC_2.2) [LSB]	write(GLIBC_2.2) [SUSv3]

writev(GLIBC_2.2) [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_2.2) [1]~~	~~fgetpos(GLIBC_2.2) [2]~~	~~fsetpos(GLIBC_2.2) [2]~~	~~putchar(GLIBC_2.2) [2]~~	~~sscanf(GLIBC_2.2) [1]~~
~~_IO_getc(GLIBC_2.2) [1]~~	~~fgets(GLIBC_2.2) [2]~~	~~ftell(GLIBC_2.2) [2]~~	~~putchar_unlocked(GLIBC_2.2) [2]~~	~~telldir(GLIBC_2.2) [2]~~
~~_IO_putc(GLIBC_2.2) [1]~~	~~fgetwc_unlocked(GLIBC_2.2) [1]~~	~~ftello(GLIBC_2.2) [2]~~	~~puts(GLIBC_2.2) [2]~~	~~tempnam(GLIBC_2.2) [2]~~
~~_IO_puts(GLIBC_2.2) [1]~~	~~fileno(GLIBC_2.2) [2]~~	~~fwrite(GLIBC_2.2) [2]~~	~~putw(GLIBC_2.2) [3]~~	~~ungetc(GLIBC_2.2) [2]~~
~~asprintf(GLIBC_2.2) [1]~~	~~flockfile(GLIBC_2.2) [2]~~	~~getc(GLIBC_2.2) [2]~~	~~remove(GLIBC_2.2) [2]~~	~~vasprintf(GLIBC_2.2) [1]~~
~~clearerr(GLIBC_2.2) [2]~~	~~fopen(GLIBC_2.2) [2]~~	~~getc_unlocked(GLIBC_2.2) [2]~~	~~rewind(GLIBC_2.2) [2]~~	~~vdprintf(GLIBC_2.2) [1]~~
~~ctermid(GLIBC_2.2) [2]~~	~~fprintf(GLIBC_2.2) [2]~~	~~getchar(GLIBC_2.2) [2]~~	~~rewinddir(GLIBC_2.2) [2]~~	~~vfprintf(GLIBC_2.2) [2]~~
~~fclose(GLIBC_2.2) [2]~~	~~fputc(GLIBC_2.2) [2]~~	~~getchar_unlocked(GLIBC_2.2) [2]~~	~~scanf(GLIBC_2.2) [1]~~	~~vprintf(GLIBC_2.2) [2]~~
~~fdopen(GLIBC_2.2) [2]~~	~~fputs(GLIBC_2.2) [2]~~	~~getw(GLIBC_2.2) [3]~~	~~seekdir(GLIBC_2.2) [2]~~	~~vsnprintf(GLIBC_2.2) [2]~~
~~feof(GLIBC_2.2) [2]~~	~~fread(GLIBC_2.2) [2]~~	~~pclose(GLIBC_2.2) [2]~~	~~setbuf(GLIBC_2.2) [2]~~	~~vsprintf(GLIBC_2.2) [2]~~
~~ferror(GLIBC_2.2) [2]~~	~~freopen(GLIBC_2.2) [2]~~	~~popen(GLIBC_2.2) [2]~~	~~setbuffer(GLIBC_2.2) [1]~~	
~~fflush(GLIBC_2.2) [2]~~	~~fscanf(GLIBC_2.2) [1]~~	~~printf(GLIBC_2.2) [2]~~	~~setvbuf(GLIBC_2.2) [2]~~	
~~fflush_unlocked(GLIBC_2.2) [1]~~	~~fseek(GLIBC_2.2) [2]~~	~~putc(GLIBC_2.2) [2]~~	~~snprintf(GLIBC_2.2) [2]~~	
~~fgetc(GLIBC_2.2) [2]~~	~~fseeko(GLIBC_2.2) [2]~~	~~putc_unlocked(GLIBC_2.2)~~	~~sprintf(GLIBC_2.2) [2]~~	

		~~[2]~~		

~~*Referenced Specification(s)*~~

~~**[1].**~~

_IO_feof(GLIBC_2.2) [LSB]	_IO_getc(GLIBC_2.2) [LSB]	_IO_putc(GLIBC_2.2) [LSB]	_IO_puts(GLIBC_2.2) [LSB]
asprintf(GLIBC_2.2) [LSB]	clearerr(GLIBC_2.2) [SUSv3]	ctermid(GLIBC_2.2) [SUSv3]	fclose(GLIBC_2.2) [SUSv3]
fdopen(GLIBC_2.2) [SUSv3]	feof(GLIBC_2.2) [SUSv3]	ferror(GLIBC_2.2) [SUSv3]	fflush(GLIBC_2.2) [SUSv3]
fflush_unlocked(GLIBC_2.2) [LSB]	fgetc(GLIBC_2.2) [SUSv3]	fgetpos(GLIBC_2.2) [SUSv3]	fgets(GLIBC_2.2) [SUSv3]
fgetwc_unlocked(GLIBC_2.2) [LSB]	fileno(GLIBC_2.2) [SUSv3]	flockfile(GLIBC_2.2) [SUSv3]	fopen(GLIBC_2.2) [SUSv3]
fprintf(GLIBC_2.2) [SUSv3]	fputc(GLIBC_2.2) [SUSv3]	fputs(GLIBC_2.2) [SUSv3]	fread(GLIBC_2.2) [SUSv3]
freopen(GLIBC_2.2) [SUSv3]	fscanf(GLIBC_2.2) [LSB]	fseek(GLIBC_2.2) [SUSv3]	fseeko(GLIBC_2.2) [SUSv3]
fsetpos(GLIBC_2.2) [SUSv3]	ftell(GLIBC_2.2) [SUSv3]	ftello(GLIBC_2.2) [SUSv3]	fwrite(GLIBC_2.2) [SUSv3]
getc(GLIBC_2.2) [SUSv3]	getc_unlocked(GLIBC_2.2) [SUSv3]	getchar(GLIBC_2.2) [SUSv3]	getchar_unlocked(GLIBC_2.2) [SUSv3]
getw(GLIBC_2.2) [SUSv2]	pclose(GLIBC_2.2) [SUSv3]	popen(GLIBC_2.2) [SUSv3]	printf(GLIBC_2.2) [SUSv3]
putc(GLIBC_2.2) [SUSv3]	putc_unlocked(GLIBC_2.2) [SUSv3]	putchar(GLIBC_2.2) [SUSv3]	putchar_unlocked(GLIBC_2.2) [SUSv3]
puts(GLIBC_2.2) [SUSv3]	putw(GLIBC_2.2) [SUSv2]	remove(GLIBC_2.2) [SUSv3]	rewind(GLIBC_2.2) [SUSv3]
rewinddir(GLIBC_2.2) [SUSv3]	scanf(GLIBC_2.2) [LSB]	seekdir(GLIBC_2.2) [SUSv3]	setbuf(GLIBC_2.2) [SUSv3]
setbuffer(GLIBC_2.2) [LSB]	setvbuf(GLIBC_2.2) [SUSv3]	snprintf(GLIBC_2.2) [SUSv3]	sprintf(GLIBC_2.2) [SUSv3]
sscanf(GLIBC_2.2) [LSB]	telldir(GLIBC_2.2) [SUSv3]	tempnam(GLIBC_2.2) [SUSv3]	ungetc(GLIBC_2.2) [SUSv3]
vasprintf(GLIBC_2.2) [LSB]	vdprintf(GLIBC_2.2) [LSB]	vfprintf(GLIBC_2.2) [SUSv3]	vprintf(GLIBC_2.2) [SUSv3]
vsnprintf(GLIBC_2.2) [SUSv3]	vsprintf(GLIBC_2.2) [SUSv3]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard I/O specified in ~~this specification~~Table 11-5

~~[2]. ISO POSIX (2003)~~

~~[3]. SUSv2~~

~~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_2.2) [1]~~	~~stdin(GLIBC_2.2) [1]~~	~~stdout(GLIBC_2.2) [1]~~		

~~*Referenced Specification(s)*~~

~~[1]. ISO POSIX (2003)~~

stderr(GLIBC_2.2) [SUSv3]	stdin(GLIBC_2.2) [SUSv3]	stdout(GLIBC_2.2) [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_2.2) [1]~~	~~sigaction(GLIBC_2.2) [2]~~	~~sighold(GLIBC_2.2) [2]~~	~~sigorset(GLIBC_2.2) [1]~~	~~sigset(GLIBC_2.2) [2]~~
~~__libc_current_sigrtmin(GLIBC_2.2) [1]~~	~~sigaddset(GLIBC_2.2) [2]~~	~~sigignore(GLIBC_2.2) [2]~~	~~sigpause(GLIBC_2.2) [2]~~	~~sigsuspend(GLIBC_2.2) [2]~~
~~__sigsetjmp(GLIBC_2.2) [1]~~	~~sigaltstack(GLIBC_2.2) [2]~~	~~siginterrupt(GLIBC_2.2) [2]~~	~~sigpending(GLIBC_2.2) [2]~~	~~sigtimedwait(GLIBC_2.2) [2]~~
~~__sysv_signal(GLIBC_2.2) [1]~~	~~sigandset(GLIBC_2.2) [1]~~	~~sigisemptyset(GLIBC_2.2) [1]~~	~~sigprocmask(GLIBC_2.2) [2]~~	~~sigwait(GLIBC_2.2) [2]~~
~~bsd_signal(GLIBC_2.2) [2]~~	~~sigdelset(GLIBC_2.2) [2]~~	~~sigismember(GLIBC_2.2) [2]~~	~~sigqueue(GLIBC_2.2) [2]~~	~~sigwaitinfo(GLIBC_2.2) [2]~~
~~psignal(GLIBC_2.2) [1]~~	~~sigemptyset(GLIBC_2.2) [2]~~	~~siglongjmp(GLIBC_2.2) [2]~~	~~sigrelse(GLIBC_2.2) [2]~~	
~~raise(GLIBC_2.2) [2]~~	~~sigfillset(GLIBC_2.2) [2]~~	~~signal(GLIBC_2.2) [2]~~	~~sigreturn(GLIBC_2.2) [1]~~	

~~*Referenced Specification(s)*~~

~~[1].~~

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

An LSB conforming implementation shall provide the architecture specific data interfaces for Signal Handling specified in ~~this specification~~Table 11-7

~~[2]. ISO POSIX (2003)~~

~~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) [1]~~				

~~*Referenced Specification(s)*~~

~~[1]. this specification~~

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

~~*Referenced Specification(s)*~~

~~[1].~~

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

An LSB conforming implementation shall provide the architecture specific data interfaces for Localization Functions specified in ~~this specification~~Table 11-9

~~[2]. ISO POSIX (2003)~~

~~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_2.2) [1]~~				

~~*Referenced Specification(s)*~~

~~[1]. this specification~~

_nl_msg_cat_cntr(GLIBC_2.2) [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_2.2) [1]~~	~~gethostname(GLIBC_2.2) [2]~~	~~if_nameindex(GLIBC_2.2) [2]~~	~~send(GLIBC_2.2) [2]~~	~~socket(GLIBC_2.2) [2]~~
~~accept(GLIBC_2.2) [2]~~	~~getpeername(GLIBC_2.2) [2]~~	~~if_nametoindex(GLIBC_2.2) [2]~~	~~sendmsg(GLIBC_2.2) [2]~~	~~socketpair(GLIBC_2.2) [2]~~
~~bind(GLIBC_2.2) [2]~~	~~getsockname(GLIBC_2.2) [2]~~	~~listen(GLIBC_2.2) [2]~~	~~sendto(GLIBC_2.2) [2]~~	
~~bindresvport(GLIBC_2.2) [1]~~	~~getsockopt(GLIBC_2.2) [1]~~	~~recv(GLIBC_2.2) [2]~~	~~setsockopt(GLIBC_2.2) [1]~~	
~~connect(GLIBC_2.2) [2]~~	~~if_freenameindex(GLIBC_2.2) [2]~~	~~recvfrom(GLIBC_2.2) [2]~~	~~shutdown(GLIBC_2.2) [2]~~	
~~gethostid(GLIBC_2.2) [2]~~	~~if_indextoname(GLIBC_2.2) [2]~~	~~recvmsg(GLIBC_2.2) [2]~~	~~sockatmark(GLIBC_2.2.4) [2]~~	

~~Referenced Specification(s)~~

~~[1]. this specification~~

~~[2]. ISO POSIX (2003)~~

__h_errno_location(GLIBC_2.2) [LSB]	accept(GLIBC_2.2) [SUSv3]	bind(GLIBC_2.2) [SUSv3]	bindresvport(GLIBC_2.2) [LSB]
connect(GLIBC_2.2) [SUSv3]	gethostid(GLIBC_2.2) [SUSv3]	gethostname(GLIBC_2.2) [SUSv3]	getpeername(GLIBC_2.2) [SUSv3]
getsockname(GLIBC_2.2) [SUSv3]	getsockopt(GLIBC_2.2) [LSB]	if_freenameindex(GLIBC_2.2) [SUSv3]	if_indextoname(GLIBC_2.2) [SUSv3]
if_nameindex(GLIBC_2.2) [SUSv3]	if_nametoindex(GLIBC_2.2) [SUSv3]	listen(GLIBC_2.2) [SUSv3]	recv(GLIBC_2.2) [SUSv3]
recvfrom(GLIBC_2.2) [SUSv3]	recvmsg(GLIBC_2.2) [SUSv3]	send(GLIBC_2.2) [SUSv3]	sendmsg(GLIBC_2.2) [SUSv3]
sendto(GLIBC_2.2) [SUSv3]	setsockopt(GLIBC_2.2) [LSB]	shutdown(GLIBC_2.2) [SUSv3]	sockatmark(GLIBC_2.2.4) [SUSv3]

socket(GLIBC_2.2) [SUSv3]	socketpair(GLIBC_2.2) [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_2.2) [1]~~	~~mbsinit(GLIBC_2.2) [2]~~	~~vwscanf(GLIBC_2.2) [1]~~	~~wcsnlen(GLIBC_2.2) [1]~~	~~wcstoumax(GLIBC_2.2) [2]~~
~~__wcstof_internal(GLIBC_2.2) [1]~~	~~mbsnrtowcs(GLIBC_2.2) [1]~~	~~wcpcpy(GLIBC_2.2) [1]~~	~~wcsnrtombs(GLIBC_2.2) [1]~~	~~wcstouq(GLIBC_2.2) [1]~~
~~__wcstol_internal(GLIBC_2.2) [1]~~	~~mbsrtowcs(GLIBC_2.2) [2]~~	~~wcpncpy(GLIBC_2.2) [1]~~	~~wcspbrk(GLIBC_2.2) [2]~~	~~wcswcs(GLIBC_2.2) [2]~~
~~__wcstold_internal(GLIBC_2.2) [1]~~	~~mbstowcs(GLIBC_2.2) [2]~~	~~wcrtomb(GLIBC_2.2) [2]~~	~~wcsrchr(GLIBC_2.2) [2]~~	~~wcswidth(GLIBC_2.2) [2]~~
~~__wcstoul_internal(GLIBC_2.2) [1]~~	~~mbtowc(GLIBC_2.2) [2]~~	~~wcscasecmp(GLIBC_2.2) [1]~~	~~wcsrtombs(GLIBC_2.2) [2]~~	~~wcsxfrm(GLIBC_2.2) [2]~~
~~btowc(GLIBC_2.2) [2]~~	~~putwc(GLIBC_2.2) [2]~~	~~wcscat(GLIBC_2.2) [2]~~	~~wcsspn(GLIBC_2.2) [2]~~	~~wctob(GLIBC_2.2) [2]~~
~~fgetwc(GLIBC_2.2) [2]~~	~~putwchar(GLIBC_2.2) [2]~~	~~wcschr(GLIBC_2.2) [2]~~	~~wcsstr(GLIBC_2.2) [2]~~	~~wctomb(GLIBC_2.2) [2]~~
~~fgetws(GLIBC_2.2) [2]~~	~~swprintf(GLIBC_2.2) [2]~~	~~wcscmp(GLIBC_2.2) [2]~~	~~wcstod(GLIBC_2.2) [2]~~	~~wctrans(GLIBC_2.2) [2]~~
~~fputwc(GLIBC_2.2) [2]~~	~~swscanf(GLIBC_2.2) [1]~~	~~wcscoll(GLIBC_2.2) [2]~~	~~wcstof(GLIBC_2.2) [2]~~	~~wctype(GLIBC_2.2) [2]~~
~~fputws(GLIBC_2.2) [2]~~	~~towctrans(GLIBC_2.2) [2]~~	~~wcscpy(GLIBC_2.2) [2]~~	~~wcstoimax(GLIBC_2.2) [2]~~	~~wcwidth(GLIBC_2.2) [2]~~
~~fwide(GLIBC_2.2) [2]~~	~~towlower(GLIBC_2.2) [2]~~	~~wcscspn(GLIBC_2.2) [2]~~	~~wcstok(GLIBC_2.2) [2]~~	~~wmemchr(GLIBC_2.2) [2]~~
~~fwprintf(GLIBC_2.2) [2]~~	~~towupper(GLIBC_2.2) [2]~~	~~wcsdup(GLIBC_2.2) [1]~~	~~wcstol(GLIBC_2.2) [2]~~	~~wmemcmp(GLIBC_2.2) [2]~~
~~fwscanf(GLIBC_2.2) [1]~~	~~ungetwc(GLIBC_2.2) [2]~~	~~wcsftime(GLIBC_2.2) [2]~~	~~wcstold(GLIBC_2.2) [2]~~	~~wmemcpy(GLIBC_2.2) [2]~~
~~getwc(GLIBC_2.2) [2]~~	~~vfwprintf(GLIBC_2.2) [2]~~	~~wcslen(GLIBC_2.2) [2]~~	~~wcstoll(GLIBC_2.2) [2]~~	~~wmemmove(GLIBC_2.2)~~

				~~[2]~~
~~getwchar(GLIBC_2.2) [2]~~	~~vfwscanf(GLIBC_2.2) [1]~~	~~wcsncasecmp (GLIBC_2.2) [1]~~	~~wcstombs(GLIBC_2.2) [2]~~	~~wmemset(GLIBC_2.2) [2]~~
~~mblen(GLIBC_2.2) [2]~~	~~vswprintf(GLIBC_2.2) [2]~~	~~wcsncat(GLIBC_2.2) [2]~~	~~wcstoq(GLIBC_2.2) [1]~~	~~wprintf(GLIBC_2.2) [2]~~
~~mbrlen(GLIBC_2.2) [2]~~	~~vswscanf(GLIBC_2.2) [1]~~	~~wcsncmp(GLIBC_2.2) [2]~~	~~wcstoul(GLIBC_2.2) [2]~~	~~wscanf(GLIBC_2.2) [1]~~
~~mbrtowc(GLIBC_2.2) [2]~~	~~vwprintf(GLIBC_2.2) [2]~~	~~wcsncpy(GLIBC_2.2) [2]~~	~~wcstoull(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

~~**[2].** ISO POSIX (2003)~~

__wcstod_internal(GLIBC_2.2) [LSB]	__wcstof_internal(GLIBC_2.2) [LSB]	__wcstol_internal(GLIBC_2.2) [LSB]	__wcstold_internal(GLIBC_2.2) [LSB]
__wcstoul_internal(GLIBC_2.2) [LSB]	btowc(GLIBC_2.2) [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_2.2) [SUSv3]
mbrlen(GLIBC_2.2) [SUSv3]	mbrtowc(GLIBC_2.2) [SUSv3]	mbsinit(GLIBC_2.2) [SUSv3]	mbsnrtowcs(GLIBC_2.2) [LSB]
mbsrtowcs(GLIBC_2.2) [SUSv3]	mbstowcs(GLIBC_2.2) [SUSv3]	mbtowc(GLIBC_2.2) [SUSv3]	putwc(GLIBC_2.2) [SUSv3]
putwchar(GLIBC_2.2) [SUSv3]	swprintf(GLIBC_2.2) [SUSv3]	swscanf(GLIBC_2.2) [LSB]	towctrans(GLIBC_2.2) [SUSv3]
towlower(GLIBC_2.2) [SUSv3]	towupper(GLIBC_2.2) [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_2.2) [LSB]	wcpncpy(GLIBC_2.2) [LSB]	wcrtomb(GLIBC_2.2) [SUSv3]
wcscasecmp(GLIBC_2.2) [LSB]	wcscat(GLIBC_2.2) [SUSv3]	wcschr(GLIBC_2.2) [SUSv3]	wcscmp(GLIBC_2.2) [SUSv3]
wcscoll(GLIBC_2.2) [SUSv3]	wcscpy(GLIBC_2.2) [SUSv3]	wcscspn(GLIBC_2.2) [SUSv3]	wcsdup(GLIBC_2.2) [LSB]
wcsftime(GLIBC_2.2) [SUSv3]	wcslen(GLIBC_2.2) [SUSv3]	wcsncasecmp(GLIBC_2.2) [LSB]	wcsncat(GLIBC_2.2) [SUSv3]

wcsncmp(GLIBC_2.2) [SUSv3]	wcsncpy(GLIBC_2.2) [SUSv3]	wcsnlen(GLIBC_2.2) [LSB]	wcsnrtombs(GLIBC_2.2) [LSB]
wcspbrk(GLIBC_2.2) [SUSv3]	wcsrchr(GLIBC_2.2) [SUSv3]	wcsrtombs(GLIBC_2.2) [SUSv3]	wcsspn(GLIBC_2.2) [SUSv3]
wcsstr(GLIBC_2.2) [SUSv3]	wcstod(GLIBC_2.2) [SUSv3]	wcstof(GLIBC_2.2) [SUSv3]	wcstoimax(GLIBC_2.2) [SUSv3]
wcstok(GLIBC_2.2) [SUSv3]	wcstol(GLIBC_2.2) [SUSv3]	wcstold(GLIBC_2.2) [SUSv3]	wcstoll(GLIBC_2.2) [SUSv3]
wcstombs(GLIBC_2.2) [SUSv3]	wcstoq(GLIBC_2.2) [LSB]	wcstoul(GLIBC_2.2) [SUSv3]	wcstoull(GLIBC_2.2) [SUSv3]
wcstoumax(GLIBC_2.2) [SUSv3]	wcstouq(GLIBC_2.2) [LSB]	wcswcs(GLIBC_2.2) [SUSv3]	wcswidth(GLIBC_2.2) [SUSv3]
wcsxfrm(GLIBC_2.2) [SUSv3]	wctob(GLIBC_2.2) [SUSv3]	wctomb(GLIBC_2.2) [SUSv3]	wctrans(GLIBC_2.2) [SUSv3]
wctype(GLIBC_2.2) [SUSv3]	wcwidth(GLIBC_2.2) [SUSv3]	wmemchr(GLIBC_2.2) [SUSv3]	wmemcmp(GLIBC_2.2) [SUSv3]
wmemcpy(GLIBC_2.2) [SUSv3]	wmemmove(GLIBC_2.2) [SUSv3]	wmemset(GLIBC_2.2) [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_2.2) [1]~~	~~bzero(GLIBC_2.2) [2]~~	~~strcasestr(GLIBC_2.2) [1]~~	~~strncat(GLIBC_2.2) [2]~~	~~strtok(GLIBC_2.2) [2]~~
~~__rawmemchr(GLIBC_2.2) [1]~~	~~ffs(GLIBC_2.2) [2]~~	~~strcat(GLIBC_2.2) [2]~~	~~strncmp(GLIBC_2.2) [2]~~	~~strtok_r(GLIBC_2.2) [2]~~
~~__stpcpy(GLIBC_2.2) [1]~~	~~index(GLIBC_2.2) [2]~~	~~strchr(GLIBC_2.2) [2]~~	~~strncpy(GLIBC_2.2) [2]~~	~~strtold(GLIBC_2.2) [2]~~
~~__strdup(GLIBC_2.2) [1]~~	~~memccpy(GLIBC_2.2) [2]~~	~~strcmp(GLIBC_2.2) [2]~~	~~strndup(GLIBC_2.2) [1]~~	~~strtoll(GLIBC_2.2) [2]~~
~~__strtod_internal(GLIBC_2.2) [1]~~	~~memchr(GLIBC_2.2) [2]~~	~~strcoll(GLIBC_2.2) [2]~~	~~strnlen(GLIBC_2.2) [1]~~	~~strtoq(GLIBC_2.2) [1]~~
~~__strtof_inter~~	~~memcmp(GLI~~	~~strcpy(GLIBC~~	~~strpbrk(GLIB~~	~~strtoull(GLIB~~

~~nal(GLIBC_2.2) [1]~~	~~BC_2.2) [2]~~	~~_2.2) [2]~~	~~C_2.2) [2]~~	~~C_2.2) [2]~~
~~__strtok_r(GLIBC_2.2) [1]~~	~~memcpy(GLIBC_2.2) [2]~~	~~strcspn(GLIBC_2.2) [2]~~	~~strptime(GLIBC_2.2) [1]~~	~~strtoumax(GLIBC_2.2) [2]~~
~~__strtol_internal(GLIBC_2.2) [1]~~	~~memmove(GLIBC_2.2) [2]~~	~~strdup(GLIBC_2.2) [2]~~	~~strrchr(GLIBC_2.2) [2]~~	~~strtouq(GLIBC_2.2) [1]~~
~~__strtold_internal(GLIBC_2.2) [1]~~	~~memrchr(GLIBC_2.2) [1]~~	~~strerror(GLIBC_2.2) [2]~~	~~strsep(GLIBC_2.2) [1]~~	~~strxfrm(GLIBC_2.2) [2]~~
~~__strtoll_internal(GLIBC_2.2) [1]~~	~~memset(GLIBC_2.2) [2]~~	~~strerror_r(GLIBC_2.2) [1]~~	~~strsignal(GLIBC_2.2) [1]~~	~~swab(GLIBC_2.2) [2]~~
~~__strtoul_internal(GLIBC_2.2) [1]~~	~~rindex(GLIBC_2.2) [2]~~	~~strfmon(GLIBC_2.2) [2]~~	~~strspn(GLIBC_2.2) [2]~~	
~~__strtoull_internal(GLIBC_2.2) [1]~~	~~stpcpy(GLIBC_2.2) [1]~~	~~strftime(GLIBC_2.2) [2]~~	~~strstr(GLIBC_2.2) [2]~~	
~~bcmp(GLIBC_2.2) [2]~~	~~stpncpy(GLIBC_2.2) [1]~~	~~strlen(GLIBC_2.2) [2]~~	~~strtof(GLIBC_2.2) [2]~~	
~~bcopy(GLIBC_2.2) [2]~~	~~strcasecmp(GLIBC_2.2) [2]~~	~~strncasecmp(GLIBC_2.2) [2]~~	~~strtoimax(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

~~**[2].** ISO POSIX (2003)~~

__mempcpy(GLIBC_2.2) [LSB]	__rawmemchr(GLIBC_2.2) [LSB]	__stpcpy(GLIBC_2.2) [LSB]	__strdup(GLIBC_2.2) [LSB]
__strtod_internal(GLIBC_2.2) [LSB]	__strtof_internal(GLIBC_2.2) [LSB]	__strtok_r(GLIBC_2.2) [LSB]	__strtol_internal(GLIBC_2.2) [LSB]
__strtold_internal(GLIBC_2.2) [LSB]	__strtoll_internal(GLIBC_2.2) [LSB]	__strtoul_internal(GLIBC_2.2) [LSB]	__strtoull_internal(GLIBC_2.2) [LSB]
bcmp(GLIBC_2.2) [SUSv3]	bcopy(GLIBC_2.2) [SUSv3]	bzero(GLIBC_2.2) [SUSv3]	ffs(GLIBC_2.2) [SUSv3]
index(GLIBC_2.2) [SUSv3]	memccpy(GLIBC_2.2) [SUSv3]	memchr(GLIBC_2.2) [SUSv3]	memcmp(GLIBC_2.2) [SUSv3]
memcpy(GLIBC_2.2) [SUSv3]	memmove(GLIBC_2.2) [SUSv3]	memrchr(GLIBC_2.2) [LSB]	memset(GLIBC_2.2) [SUSv3]
rindex(GLIBC_2.2) [SUSv3]	stpcpy(GLIBC_2.2) [LSB]	stpncpy(GLIBC_2.2) [LSB]	strcasecmp(GLIBC_2.2) [SUSv3]
strcasestr(GLIBC_	strcat(GLIBC_2.2)	strchr(GLIBC_2.2)	strcmp(GLIBC_2.2

2.2) [LSB]	[SUSv3]	[SUSv3]	) [SUSv3]
strcoll(GLIBC_2.2) [SUSv3]	strcpy(GLIBC_2.2) [SUSv3]	strcspn(GLIBC_2.2) [SUSv3]	strdup(GLIBC_2.2) [SUSv3]
strerror(GLIBC_2.2) [SUSv3]	strerror_r(GLIBC_2.2) [LSB]	strfmon(GLIBC_2.2) [SUSv3]	strftime(GLIBC_2.2) [SUSv3]
strlen(GLIBC_2.2) [SUSv3]	strncasecmp(GLIBC_2.2) [SUSv3]	strncat(GLIBC_2.2) [SUSv3]	strncmp(GLIBC_2.2) [SUSv3]
strncpy(GLIBC_2.2) [SUSv3]	strndup(GLIBC_2.2) [LSB]	strnlen(GLIBC_2.2) [LSB]	strpbrk(GLIBC_2.2) [SUSv3]
strptime(GLIBC_2.2) [LSB]	strrchr(GLIBC_2.2) [SUSv3]	strsep(GLIBC_2.2) [LSB]	strsignal(GLIBC_2.2) [LSB]
strspn(GLIBC_2.2) [SUSv3]	strstr(GLIBC_2.2) [SUSv3]	strtof(GLIBC_2.2) [SUSv3]	strtoimax(GLIBC_2.2) [SUSv3]
strtok(GLIBC_2.2) [SUSv3]	strtok_r(GLIBC_2.2) [SUSv3]	strtold(GLIBC_2.2) [SUSv3]	strtoll(GLIBC_2.2) [SUSv3]
strtoq(GLIBC_2.2) [LSB]	strtoull(GLIBC_2.2) [SUSv3]	strtoumax(GLIBC_2.2) [SUSv3]	strtouq(GLIBC_2.2) [LSB]
strxfrm(GLIBC_2.2) [SUSv3]	swab(GLIBC_2.2) [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_2.2) [1]~~	~~msgrcv(GLIBC_2.2) [1]~~	~~semget(GLIBC_2.2) [1]~~	~~shmctl(GLIBC_2.2) [1]~~	
~~msgctl(GLIBC_2.2) [1]~~	~~msgsnd(GLIBC_2.2) [1]~~	~~semop(GLIBC_2.2) [1]~~	~~shmdt(GLIBC_2.2) [1]~~	
~~msgget(GLIBC_2.2) [1]~~	~~semctl(GLIBC_2.2) [1]~~	~~shmat(GLIBC_2.2) [1]~~	~~shmget(GLIBC_2.2) [1]~~	

~~*Referenced Specification(s)*~~

~~**[1].** ISO POSIX (2003)~~

ftok(GLIBC_2.2) [SUSv3]	msgctl(GLIBC_2.2) [SUSv3]	msgget(GLIBC_2.2) [SUSv3]	msgrcv(GLIBC_2.2) [SUSv3]
msgsnd(GLIBC_2.2) [SUSv3]	semctl(GLIBC_2.2) [SUSv3]	semget(GLIBC_2.2) [SUSv3]	semop(GLIBC_2.2) [SUSv3]
shmat(GLIBC_2.2) [SUSv3]	shmctl(GLIBC_2.2) [SUSv3]	shmdt(GLIBC_2.2) [SUSv3]	shmget(GLIBC_2.2) [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_2.2) [1]~~	~~regerror(GLIBC_2.2) [1]~~	~~regexec(GLIBC_2.3.4) [2]~~	~~regfree(GLIBC_2.2) [1]~~	

~~*Referenced Specification(s)*~~

~~**[1].** ISO POSIX (2003)~~

~~**[2].** this specification~~

regcomp(GLIBC_2.2) [SUSv3]	regerror(GLIBC_2.2) [SUSv3]	regexec(GLIBC_2.3.4) [LSB]	regfree(GLIBC_2.2) [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_2.2) [1]~~	~~isdigit(GLIBC_2.2) [2]~~	~~iswalnum(GLIBC_2.2) [2]~~	~~iswlower(GLIBC_2.2) [2]~~	~~toascii(GLIBC_2.2) [2]~~
~~_tolower(GLIBC_2.2) [2]~~	~~isgraph(GLIBC_2.2) [2]~~	~~iswalpha(GLIBC_2.2) [2]~~	~~iswprint(GLIBC_2.2) [2]~~	~~tolower(GLIBC_2.2) [2]~~
~~_toupper(GLIBC_2.2) [2]~~	~~islower(GLIBC_2.2) [2]~~	~~iswblank(GLIBC_2.2) [2]~~	~~iswpunct(GLIBC_2.2) [2]~~	~~toupper(GLIBC_2.2) [2]~~
~~isalnum(GLIBC_2.2) [2]~~	~~isprint(GLIBC_2.2) [2]~~	~~iswcntrl(GLIBC_2.2) [2]~~	~~iswspace(GLIBC_2.2) [2]~~	
~~isalpha(GLIBC_2.2) [2]~~	~~ispunct(GLIBC_2.2) [2]~~	~~iswctype(GLIBC_2.2) [2]~~	~~iswupper(GLIBC_2.2) [2]~~	
~~isascii(GLIBC_2.2) [2]~~	~~isspace(GLIBC_2.2) [2]~~	~~iswdigit(GLIBC_2.2) [2]~~	~~iswxdigit(GLIBC_2.2) [2]~~	
~~iscntrl(GLIBC_2.2) [2]~~	~~isupper(GLIBC_2.2) [2]~~	~~iswgraph(GLIBC_2.2) [2]~~	~~isxdigit(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

~~**[2].** ISO POSIX (2003)~~

__ctype_get_mb_cur_max(GLIBC_2.2) [LSB]	_tolower(GLIBC_2.2) [SUSv3]	_toupper(GLIBC_2.2) [SUSv3]	isalnum(GLIBC_2.2) [SUSv3]
isalpha(GLIBC_2.2) [SUSv3]	isascii(GLIBC_2.2) [SUSv3]	iscntrl(GLIBC_2.2) [SUSv3]	isdigit(GLIBC_2.2) [SUSv3]
isgraph(GLIBC_2.2) [SUSv3]	islower(GLIBC_2.2) [SUSv3]	isprint(GLIBC_2.2) [SUSv3]	ispunct(GLIBC_2.2) [SUSv3]
isspace(GLIBC_2.2) [SUSv3]	isupper(GLIBC_2.2) [SUSv3]	iswalnum(GLIBC_2.2) [SUSv3]	iswalpha(GLIBC_2.2) [SUSv3]
iswblank(GLIBC_2.2) [SUSv3]	iswcntrl(GLIBC_2.2) [SUSv3]	iswctype(GLIBC_2.2) [SUSv3]	iswdigit(GLIBC_2.2) [SUSv3]
iswgraph(GLIBC_2.2) [SUSv3]	iswlower(GLIBC_2.2) [SUSv3]	iswprint(GLIBC_2.2) [SUSv3]	iswpunct(GLIBC_2.2) [SUSv3]
iswspace(GLIBC_2.2) [SUSv3]	iswupper(GLIBC_2.2) [SUSv3]	iswxdigit(GLIBC_2.2) [SUSv3]	isxdigit(GLIBC_2.2) [SUSv3]
toascii(GLIBC_2.2) [SUSv3]	tolower(GLIBC_2.2) [SUSv3]	toupper(GLIBC_2.2) [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_2.2) [1]~~	~~ctime(GLIBC_2.2) [2]~~	~~gmtime(GLIBC_2.2) [2]~~	~~localtime_r(GLIBC_2.2) [2]~~	~~ualarm(GLIBC_2.2) [2]~~
~~asctime(GLIBC_2.2) [2]~~	~~ctime_r(GLIBC_2.2) [2]~~	~~gmtime_r(GLIBC_2.2) [2]~~	~~mktime(GLIBC_2.2) [2]~~	
~~asctime_r(GLIBC_2.2) [2]~~	~~difftime(GLIBC_2.2) [2]~~	~~localtime(GLIBC_2.2) [2]~~	~~tzset(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].**~~

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

An LSB conforming implementation shall provide the architecture specific data interfaces for Time Manipulation specified in ~~this specification~~Table 11-17

~~[2]. ISO POSIX (2003)~~

~~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_2.2) [1]~~	~~__tzname(GLIBC_2.2) [1]~~	~~timezone(GLIBC_2.2) [2]~~		
~~__timezone(GLIBC_2.2) [1]~~	~~daylight(GLIBC_2.2) [2]~~	~~tzname(GLIBC_2.2) [2]~~		

~~*Referenced Specification(s)*~~

~~[1]. this specification~~

~~[2]. ISO POSIX (2003)~~

__daylight(GLIBC_2.2) [LSB]	__timezone(GLIBC_2.2) [LSB]	__tzname(GLIBC_2.2) [LSB]	daylight(GLIBC_2.2) [SUSv3]
timezone(GLIBC_2.2) [SUSv3]	tzname(GLIBC_2.2) [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_2.2) [1]~~	~~cfsetispeed(GLIBC_2.2) [1]~~	~~tcdrain(GLIBC_2.2) [1]~~	~~tcgetattr(GLIBC_2.2) [1]~~	~~tcsendbreak(GLIBC_2.2) [1]~~
~~cfgetospeed(GLIBC_2.2) [1]~~	~~cfsetospeed(GLIBC_2.2) [1]~~	~~tcflow(GLIBC_2.2) [1]~~	~~tcgetpgrp(GLIBC_2.2) [1]~~	~~tcsetattr(GLIBC_2.2) [1]~~
~~cfmakeraw(GLIBC_2.2) [2]~~	~~cfsetspeed(GLIBC_2.2) [2]~~	~~tcflush(GLIBC_2.2) [1]~~	~~tcgetsid(GLIBC_2.2) [1]~~	~~tcsetpgrp(GLIBC_2.2) [1]~~

~~*Referenced Specification(s)*~~

~~[1]. ISO POSIX (2003)~~

~~[2]. this specification~~

cfgetispeed(GLIBC_2.2) [SUSv3]	cfgetospeed(GLIBC_2.2) [SUSv3]	cfmakeraw(GLIBC_2.2) [LSB]	cfsetispeed(GLIBC_2.2) [SUSv3]
cfsetospeed(GLIB	cfsetspeed(GLIBC	tcdrain(GLIBC_2.	tcflow(GLIBC_2.2

C_2.2) [SUSv3]	_2.2) [LSB]	2) [SUSv3]	) [SUSv3]
tcflush(GLIBC_2.2) [SUSv3]	tcgetattr(GLIBC_2.2) [SUSv3]	tcgetpgrp(GLIBC_2.2) [SUSv3]	tcgetsid(GLIBC_2.2) [SUSv3]
tcsendbreak(GLIBC_2.2) [SUSv3]	tcsetattr(GLIBC_2.2) [SUSv3]	tcsetpgrp(GLIBC_2.2) [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_2.2) [1]~~	~~getgrgid_r(GLIBC_2.2) [1]~~	~~getprotoent(GLIBC_2.2) [1]~~	~~getservent(GLIBC_2.2) [1]~~	~~setgroups(GLIBC_2.2) [2]~~
~~endprotoent(GLIBC_2.2) [1]~~	~~getgrnam(GLIBC_2.2) [1]~~	~~getpwent(GLIBC_2.2) [1]~~	~~getutent(GLIBC_2.2) [2]~~	~~setprotoent(GLIBC_2.2) [1]~~
~~endpwent(GLIBC_2.2) [1]~~	~~getgrnam_r(GLIBC_2.2) [1]~~	~~getpwnam(GLIBC_2.2) [1]~~	~~getutent_r(GLIBC_2.2) [2]~~	~~setpwent(GLIBC_2.2) [1]~~
~~endservent(GLIBC_2.2) [1]~~	~~getgrouplist(GLIBC_2.2.4) [2]~~	~~getpwnam_r(GLIBC_2.2) [1]~~	~~getutxent(GLIBC_2.2) [1]~~	~~setservent(GLIBC_2.2) [1]~~
~~endutent(GLIBC_2.2) [3]~~	~~gethostbyaddr(GLIBC_2.2) [1]~~	~~getpwuid(GLIBC_2.2) [1]~~	~~getutxid(GLIBC_2.2) [1]~~	~~setutent(GLIBC_2.2) [2]~~
~~endutxent(GLIBC_2.2) [1]~~	~~gethostbyname(GLIBC_2.2) [1]~~	~~getpwuid_r(GLIBC_2.2) [1]~~	~~getutxline(GLIBC_2.2) [1]~~	~~setutxent(GLIBC_2.2) [1]~~
~~getgrent(GLIBC_2.2) [1]~~	~~getprotobyname(GLIBC_2.2) [1]~~	~~getservbyname(GLIBC_2.2) [1]~~	~~pututxline(GLIBC_2.2) [1]~~	~~utmpname(GLIBC_2.2) [2]~~
~~getgrgid(GLIBC_2.2) [1]~~	~~getprotobynumber(GLIBC_2.2) [1]~~	~~getservbyport(GLIBC_2.2) [1]~~	~~setgrent(GLIBC_2.2) [1]~~	

~~*Referenced Specification(s)*~~

~~**[1].** ISO POSIX (2003)~~

~~**[2].** this specification~~

~~**[3].** SUSv2~~

endgrent(GLIBC_2.2) [SUSv3]	endprotoent(GLIBC_2.2) [SUSv3]	endpwent(GLIBC_2.2) [SUSv3]	endservent(GLIBC_2.2) [SUSv3]
endutent(GLIBC_	endutxent(GLIBC	getgrent(GLIBC_2	getgrgid(GLIBC_2

2.2) [SUSv2]	_2.2) [SUSv3]	.2) [SUSv3]	.2) [SUSv3]
getgrgid_r(GLIBC_2.2) [SUSv3]	getgrnam(GLIBC_2.2) [SUSv3]	getgrnam_r(GLIBC_2.2) [SUSv3]	getgrouplist(GLIBC_2.2.4) [LSB]
gethostbyaddr(GLIBC_2.2) [SUSv3]	gethostbyname(GLIBC_2.2) [SUSv3]	getprotobyname(GLIBC_2.2) [SUSv3]	getprotobynumber(GLIBC_2.2) [SUSv3]
getprotoent(GLIBC_2.2) [SUSv3]	getpwent(GLIBC_2.2) [SUSv3]	getpwnam(GLIBC_2.2) [SUSv3]	getpwnam_r(GLIBC_2.2) [SUSv3]
getpwuid(GLIBC_2.2) [SUSv3]	getpwuid_r(GLIBC_2.2) [SUSv3]	getservbyname(GLIBC_2.2) [SUSv3]	getservbyport(GLIBC_2.2) [SUSv3]
getservent(GLIBC_2.2) [SUSv3]	getutent(GLIBC_2.2) [LSB]	getutent_r(GLIBC_2.2) [LSB]	getutxent(GLIBC_2.2) [SUSv3]
getutxid(GLIBC_2.2) [SUSv3]	getutxline(GLIBC_2.2) [SUSv3]	pututxline(GLIBC_2.2) [SUSv3]	setgrent(GLIBC_2.2) [SUSv3]
setgroups(GLIBC_2.2) [LSB]	setprotoent(GLIBC_2.2) [SUSv3]	setpwent(GLIBC_2.2) [SUSv3]	setservent(GLIBC_2.2) [SUSv3]
setutent(GLIBC_2.2) [LSB]	setutxent(GLIBC_2.2) [SUSv3]	utmpname(GLIBC_2.2) [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_2.2) [1]~~				

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

__libc_start_main(GLIBC_2.2) [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(GL~~	~~fopen64(GLIB~~	~~ftello64(GLIB~~	~~mkstemp64(G~~	~~tmpfile64(GLI~~

~~IBC_2.2) [1]~~	~~C_2.2) [2]~~	~~C_2.2) [2]~~	~~LIBC_2.2) [2]~~	~~BC_2.2) [2]~~
~~__lxstat64(GLIBC_2.2) [1]~~	~~freopen64(GLIBC_2.2) [2]~~	~~ftruncate64(GLIBC_2.2) [2]~~	~~mmap64(GLIBC_2.2) [2]~~	~~truncate64(GLIBC_2.2) [2]~~
~~__xstat64(GLIBC_2.2) [1]~~	~~fseeko64(GLIBC_2.2) [2]~~	~~ftw64(GLIBC_2.2) [2]~~	~~nftw64(GLIBC_2.3.3) [2]~~	
~~creat64(GLIBC_2.2) [2]~~	~~fsetpos64(GLIBC_2.2) [2]~~	~~getrlimit64(GLIBC_2.2) [2]~~	~~readdir64(GLIBC_2.2) [2]~~	
~~fgetpos64(GLIBC_2.2) [2]~~	~~fstatvfs64(GLIBC_2.2) [2]~~	~~lockf64(GLIBC_2.2) [2]~~	~~statvfs64(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

~~**[2].** Large File Support~~

__fxstat64(GLIBC_2.2) [LSB]	__lxstat64(GLIBC_2.2) [LSB]	__xstat64(GLIBC_2.2) [LSB]	creat64(GLIBC_2.2) [LFS]
fgetpos64(GLIBC_2.2) [LFS]	fopen64(GLIBC_2.2) [LFS]	freopen64(GLIBC_2.2) [LFS]	fseeko64(GLIBC_2.2) [LFS]
fsetpos64(GLIBC_2.2) [LFS]	fstatvfs64(GLIBC_2.2) [LFS]	ftello64(GLIBC_2.2) [LFS]	ftruncate64(GLIBC_2.2) [LFS]
ftw64(GLIBC_2.2) [LFS]	getrlimit64(GLIBC_2.2) [LFS]	lockf64(GLIBC_2.2) [LFS]	mkstemp64(GLIBC_2.2) [LFS]
mmap64(GLIBC_2.2) [LFS]	nftw64(GLIBC_2.3.3) [LFS]	readdir64(GLIBC_2.2) [LFS]	statvfs64(GLIBC_2.2) [LFS]
tmpfile64(GLIBC_2.2) [LFS]	truncate64(GLIBC_2.2) [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_2.2) [1]~~	~~dirname(GLIBC_2.2) [1]~~	~~gettimeofday(GLIBC_2.2) [1]~~	~~lrand48(GLIBC_2.2) [1]~~	~~srand(GLIBC_2.2) [1]~~
~~__assert_fail(GLIBC_2.2) [2]~~	~~div(GLIBC_2.2) [1]~~	~~glob(GLIBC_2.2) [1]~~	~~lsearch(GLIBC_2.2) [1]~~	~~srand48(GLIBC_2.2) [1]~~
~~__cxa_atexit(GLIBC_2.2) [2]~~	~~drand48(GLIBC_2.2) [1]~~	~~glob64(GLIBC_2.2) [2]~~	~~makecontext(GLIBC_2.2) [1]~~	~~srandom(GLIBC_2.2) [1]~~

~~__errno_location(GLIBC_2.2) [2]~~	~~ecvt(GLIBC_2.2) [1]~~	~~globfree(GLIBC_2.2) [1]~~	~~malloc(GLIBC_2.2) [1]~~	~~strtod(GLIBC_2.2) [1]~~
~~__fpending(GLIBC_2.2) [2]~~	~~erand48(GLIBC_2.2) [1]~~	~~globfree64(GLIBC_2.2) [2]~~	~~memmem(GLIBC_2.2) [2]~~	~~strtol(GLIBC_2.2) [1]~~
~~__getpagesize(GLIBC_2.2) [2]~~	~~err(GLIBC_2.2) [2]~~	~~grantpt(GLIBC_2.2) [1]~~	~~mkstemp(GLIBC_2.2) [1]~~	~~strtoul(GLIBC_2.2) [1]~~
~~__isinf(GLIBC_2.2) [2]~~	~~error(GLIBC_2.2) [2]~~	~~hcreate(GLIBC_2.2) [1]~~	~~mktemp(GLIBC_2.2) [1]~~	~~swapcontext(GLIBC_2.2) [1]~~
~~__isinff(GLIBC_2.2) [2]~~	~~errx(GLIBC_2.2) [2]~~	~~hdestroy(GLIBC_2.2) [1]~~	~~mrand48(GLIBC_2.2) [1]~~	~~syslog(GLIBC_2.2) [1]~~
~~__isinfl(GLIBC_2.2) [2]~~	~~fcvt(GLIBC_2.2) [1]~~	~~hsearch(GLIBC_2.2) [1]~~	~~nftw(GLIBC_2.3.3) [1]~~	~~system(GLIBC_2.2) [2]~~
~~__isnan(GLIBC_2.2) [2]~~	~~fmtmsg(GLIBC_2.2) [1]~~	~~htonl(GLIBC_2.2) [1]~~	~~nrand48(GLIBC_2.2) [1]~~	~~tdelete(GLIBC_2.2) [1]~~
~~__isnanf(GLIBC_2.2) [2]~~	~~fnmatch(GLIBC_2.2.3) [1]~~	~~htons(GLIBC_2.2) [1]~~	~~ntohl(GLIBC_2.2) [1]~~	~~tfind(GLIBC_2.2) [1]~~
~~__isnanl(GLIBC_2.2) [2]~~	~~fpathconf(GLIBC_2.2) [1]~~	~~imaxabs(GLIBC_2.2) [1]~~	~~ntohs(GLIBC_2.2) [1]~~	~~tmpfile(GLIBC_2.2) [1]~~
~~__sysconf(GLIBC_2.2) [2]~~	~~free(GLIBC_2.2) [1]~~	~~imaxdiv(GLIBC_2.2) [1]~~	~~openlog(GLIBC_2.2) [1]~~	~~tmpnam(GLIBC_2.2) [1]~~
~~_exit(GLIBC_2.2) [1]~~	~~freeaddrinfo(GLIBC_2.2) [1]~~	~~inet_addr(GLIBC_2.2) [1]~~	~~perror(GLIBC_2.2) [1]~~	~~tsearch(GLIBC_2.2) [1]~~
~~_longjmp(GLIBC_2.2) [1]~~	~~ftrylockfile(GLIBC_2.2) [1]~~	~~inet_ntoa(GLIBC_2.2) [1]~~	~~posix_memalign(GLIBC_2.2) [1]~~	~~ttyname(GLIBC_2.2) [1]~~
~~_setjmp(GLIBC_2.2) [1]~~	~~ftw(GLIBC_2.2) [1]~~	~~inet_ntop(GLIBC_2.2) [1]~~	~~posix_openpt(GLIBC_2.2.1) [1]~~	~~ttyname_r(GLIBC_2.2) [1]~~
~~a64l(GLIBC_2.2) [1]~~	~~funlockfile(GLIBC_2.2) [1]~~	~~inet_pton(GLIBC_2.2) [1]~~	~~ptsname(GLIBC_2.2) [1]~~	~~twalk(GLIBC_2.2) [1]~~
~~abort(GLIBC_2.2) [1]~~	~~gai_strerror(GLIBC_2.2) [1]~~	~~initstate(GLIBC_2.2) [1]~~	~~putenv(GLIBC_2.2) [1]~~	~~unlockpt(GLIBC_2.2) [1]~~
~~abs(GLIBC_2.2) [1]~~	~~gcvt(GLIBC_2.2) [1]~~	~~insque(GLIBC_2.2) [1]~~	~~qsort(GLIBC_2.2) [1]~~	~~unsetenv(GLIBC_2.2) [1]~~
~~atof(GLIBC_2.2) [1]~~	~~getaddrinfo(GLIBC_2.2) [1]~~	~~isatty(GLIBC_2.2) [1]~~	~~rand(GLIBC_2.2) [1]~~	~~usleep(GLIBC_2.2) [1]~~
~~atoi(GLIBC_2.2) [1]~~	~~getcwd(GLIBC_2.2) [1]~~	~~isblank(GLIBC_2.2) [1]~~	~~rand_r(GLIBC_2.2) [1]~~	~~verrx(GLIBC_2.2) [2]~~

~~atol(GLIBC_2.2) [1]~~	~~getdate(GLIBC_2.2) [1]~~	~~jrand48(GLIBC_2.2) [1]~~	~~random(GLIBC_2.2) [1]~~	~~vfscanf(GLIBC_2.2) [2]~~
~~atoll(GLIBC_2.2) [1]~~	~~getenv(GLIBC_2.2) [1]~~	~~l64a(GLIBC_2.2) [1]~~	~~realloc(GLIBC_2.2) [1]~~	~~vscanf(GLIBC_2.2) [2]~~
~~basename(GLIBC_2.2) [1]~~	~~getlogin(GLIBC_2.2) [1]~~	~~labs(GLIBC_2.2) [1]~~	~~realpath(GLIBC_2.3) [1]~~	~~vsscanf(GLIBC_2.2) [2]~~
~~bsearch(GLIBC_2.2) [1]~~	~~getlogin_r(GLIBC_2.2) [1]~~	~~lcong48(GLIBC_2.2) [1]~~	~~remque(GLIBC_2.2) [1]~~	~~vsyslog(GLIBC_2.2) [2]~~
~~calloc(GLIBC_2.2) [1]~~	~~getnameinfo(GLIBC_2.2) [1]~~	~~ldiv(GLIBC_2.2) [1]~~	~~seed48(GLIBC_2.2) [1]~~	~~warn(GLIBC_2.2) [2]~~
~~closelog(GLIBC_2.2) [1]~~	~~getopt(GLIBC_2.2) [2]~~	~~lfind(GLIBC_2.2) [1]~~	~~setenv(GLIBC_2.2) [1]~~	~~warnx(GLIBC_2.2) [2]~~
~~confstr(GLIBC_2.2) [1]~~	~~getopt_long(GLIBC_2.2) [2]~~	~~llabs(GLIBC_2.2) [1]~~	~~sethostname(GLIBC_2.2) [2]~~	~~wordexp(GLIBC_2.2.2) [1]~~
~~cuserid(GLIBC_2.2) [3]~~	~~getopt_long_only(GLIBC_2.2) [2]~~	~~lldiv(GLIBC_2.2) [1]~~	~~setlogmask(GLIBC_2.2) [1]~~	~~wordfree(GLIBC_2.2) [1]~~
~~daemon(GLIBC_2.2) [2]~~	~~getsubopt(GLIBC_2.2) [1]~~	~~longjmp(GLIBC_2.2) [1]~~	~~setstate(GLIBC_2.2) [1]~~	

~~*Referenced Specification(s)*~~

~~**[1].**~~

_Exit(GLIBC_2.2) [SUSv3]	__assert_fail(GLIBC_2.2) [LSB]	__cxa_atexit(GLIBC_2.2) [LSB]	__errno_location(GLIBC_2.2) [LSB]
__fpending(GLIBC_2.2) [LSB]	__getpagesize(GLIBC_2.2) [LSB]	__isinf(GLIBC_2.2) [LSB]	__isinff(GLIBC_2.2) [LSB]
__isinfl(GLIBC_2.2) [LSB]	__isnan(GLIBC_2.2) [LSB]	__isnanf(GLIBC_2.2) [LSB]	__isnanl(GLIBC_2.2) [LSB]
__sysconf(GLIBC_2.2) [LSB]	_exit(GLIBC_2.2) [SUSv3]	_longjmp(GLIBC_2.2) [SUSv3]	_setjmp(GLIBC_2.2) [SUSv3]
a64l(GLIBC_2.2) [SUSv3]	abort(GLIBC_2.2) [SUSv3]	abs(GLIBC_2.2) [SUSv3]	atof(GLIBC_2.2) [SUSv3]
atoi(GLIBC_2.2) [SUSv3]	atol(GLIBC_2.2) [SUSv3]	atoll(GLIBC_2.2) [SUSv3]	basename(GLIBC_2.2) [SUSv3]
bsearch(GLIBC_2.2) [SUSv3]	calloc(GLIBC_2.2) [SUSv3]	closelog(GLIBC_2.2) [SUSv3]	confstr(GLIBC_2.2) [SUSv3]
cuserid(GLIBC_2.2) [SUSv2]	daemon(GLIBC_2.2) [LSB]	dirname(GLIBC_2.2) [SUSv3]	div(GLIBC_2.2) [SUSv3]
drand48(GLIBC_2.2) [SUSv3]	ecvt(GLIBC_2.2) [SUSv3]	erand48(GLIBC_2.2) [SUSv3]	err(GLIBC_2.2) [LSB]

error(GLIBC_2.2) [LSB]	errx(GLIBC_2.2) [LSB]	fcvt(GLIBC_2.2) [SUSv3]	fmtmsg(GLIBC_2.2) [SUSv3]
fnmatch(GLIBC_2.2.3) [SUSv3]	fpathconf(GLIBC_2.2) [SUSv3]	free(GLIBC_2.2) [SUSv3]	freeaddrinfo(GLIBC_2.2) [SUSv3]
ftrylockfile(GLIBC_2.2) [SUSv3]	ftw(GLIBC_2.2) [SUSv3]	funlockfile(GLIBC_2.2) [SUSv3]	gai_strerror(GLIBC_2.2) [SUSv3]
gcvt(GLIBC_2.2) [SUSv3]	getaddrinfo(GLIBC_2.2) [SUSv3]	getcwd(GLIBC_2.2) [SUSv3]	getdate(GLIBC_2.2) [SUSv3]
getenv(GLIBC_2.2) [SUSv3]	getlogin(GLIBC_2.2) [SUSv3]	getlogin_r(GLIBC_2.2) [SUSv3]	getnameinfo(GLIBC_2.2) [SUSv3]
getopt(GLIBC_2.2) [LSB]	getopt_long(GLIBC_2.2) [LSB]	getopt_long_only(GLIBC_2.2) [LSB]	getsubopt(GLIBC_2.2) [SUSv3]
gettimeofday(GLIBC_2.2) [SUSv3]	glob(GLIBC_2.2) [SUSv3]	glob64(GLIBC_2.2) [LSB]	globfree(GLIBC_2.2) [SUSv3]
globfree64(GLIBC_2.2) [LSB]	grantpt(GLIBC_2.2) [SUSv3]	hcreate(GLIBC_2.2) [SUSv3]	hdestroy(GLIBC_2.2) [SUSv3]
hsearch(GLIBC_2.2) [SUSv3]	htonl(GLIBC_2.2) [SUSv3]	htons(GLIBC_2.2) [SUSv3]	imaxabs(GLIBC_2.2) [SUSv3]
imaxdiv(GLIBC_2.2) [SUSv3]	inet_addr(GLIBC_2.2) [SUSv3]	inet_ntoa(GLIBC_2.2) [SUSv3]	inet_ntop(GLIBC_2.2) [SUSv3]
inet_pton(GLIBC_2.2) [SUSv3]	initstate(GLIBC_2.2) [SUSv3]	insque(GLIBC_2.2) [SUSv3]	isatty(GLIBC_2.2) [SUSv3]
isblank(GLIBC_2.2) [SUSv3]	jrand48(GLIBC_2.2) [SUSv3]	l64a(GLIBC_2.2) [SUSv3]	labs(GLIBC_2.2) [SUSv3]
lcong48(GLIBC_2.2) [SUSv3]	ldiv(GLIBC_2.2) [SUSv3]	lfind(GLIBC_2.2) [SUSv3]	llabs(GLIBC_2.2) [SUSv3]
lldiv(GLIBC_2.2) [SUSv3]	longjmp(GLIBC_2.2) [SUSv3]	lrand48(GLIBC_2.2) [SUSv3]	lsearch(GLIBC_2.2) [SUSv3]
makecontext(GLIBC_2.2) [SUSv3]	malloc(GLIBC_2.2) [SUSv3]	memmem(GLIBC_2.2) [LSB]	mkstemp(GLIBC_2.2) [SUSv3]
mktemp(GLIBC_2.2) [SUSv3]	mrand48(GLIBC_2.2) [SUSv3]	nftw(GLIBC_2.3.3) [SUSv3]	nrand48(GLIBC_2.2) [SUSv3]
ntohl(GLIBC_2.2) [SUSv3]	ntohs(GLIBC_2.2) [SUSv3]	openlog(GLIBC_2.2) [SUSv3]	perror(GLIBC_2.2) [SUSv3]
posix_memalign(GLIBC_2.2) [SUSv3]	posix_openpt(GLIBC_2.2.1) [SUSv3]	ptsname(GLIBC_2.2) [SUSv3]	putenv(GLIBC_2.2) [SUSv3]
qsort(GLIBC_2.2) [SUSv3]	rand(GLIBC_2.2) [SUSv3]	rand_r(GLIBC_2.2) [SUSv3]	random(GLIBC_2.2) [SUSv3]
realloc(GLIBC_2.2) [SUSv3]	realpath(GLIBC_2.3) [SUSv3]	remque(GLIBC_2.2) [SUSv3]	seed48(GLIBC_2.2) [SUSv3]

setenv(GLIBC_2.2) [SUSv3]	sethostname(GLIBC_2.2) [LSB]	setlogmask(GLIBC_2.2) [SUSv3]	setstate(GLIBC_2.2) [SUSv3]
srand(GLIBC_2.2) [SUSv3]	srand48(GLIBC_2.2) [SUSv3]	srandom(GLIBC_2.2) [SUSv3]	strtod(GLIBC_2.2) [SUSv3]
strtol(GLIBC_2.2) [SUSv3]	strtoul(GLIBC_2.2) [SUSv3]	swapcontext(GLIBC_2.2) [SUSv3]	syslog(GLIBC_2.2) [SUSv3]
system(GLIBC_2.2) [LSB]	tdelete(GLIBC_2.2) [SUSv3]	tfind(GLIBC_2.2) [SUSv3]	tmpfile(GLIBC_2.2) [SUSv3]
tmpnam(GLIBC_2.2) [SUSv3]	tsearch(GLIBC_2.2) [SUSv3]	ttyname(GLIBC_2.2) [SUSv3]	ttyname_r(GLIBC_2.2) [SUSv3]
twalk(GLIBC_2.2) [SUSv3]	unlockpt(GLIBC_2.2) [SUSv3]	unsetenv(GLIBC_2.2) [SUSv3]	usleep(GLIBC_2.2) [SUSv3]
verrx(GLIBC_2.2) [LSB]	vfscanf(GLIBC_2.2) [LSB]	vscanf(GLIBC_2.2) [LSB]	vsscanf(GLIBC_2.2) [LSB]
vsyslog(GLIBC_2.2) [LSB]	warn(GLIBC_2.2) [LSB]	warnx(GLIBC_2.2) [LSB]	wordexp(GLIBC_2.2.2) [SUSv3]
wordfree(GLIBC_2.2) [SUSv3]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard Library specified in ~~ISO POSIX (2003)~~Table 11-23

~~[2]. this specification~~

~~[3]. SUSv2~~

~~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_2.2) [1]~~	~~_sys_errlist(GLIBC_2.3) [1]~~	~~getdate_err(GLIBC_2.2) [2]~~	~~opterr(GLIBC_2.2) [2]~~	~~optopt(GLIBC_2.2) [2]~~
~~_environ(GLIBC_2.2) [1]~~	~~environ(GLIBC_2.2) [2]~~	~~optarg(GLIBC_2.2) [2]~~	~~optind(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~[1]. this specification~~

~~[2]. ISO POSIX (2003)~~

__environ(GLIBC_2.2) [LSB]	_environ(GLIBC_2.2) [LSB]	_sys_errlist(GLIBC_2.3) [LSB]	environ(GLIBC_2.2) [SUSv3]
getdate_err(GLIBC_2.2) [SUSv3]	optarg(GLIBC_2.2) [SUSv3]	opterr(GLIBC_2.2) [SUSv3]	optind(GLIBC_2.2) [SUSv3]
optopt(GLIBC_2.2) [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.

~~These definitions are intended to supplement those provided in~~ Where an interface is defined as requiring a particular system header file all of the ~~referenced underlying~~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 ~~ISO/IEC 9899~~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 arpa/inet.h

```

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

11.3.2 assert.h

```

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

11.3.3 ctype.h

```

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

11.3.4 dirent.h

```

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

11.3.5 err.h

```

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

11.3.6 errno.h

```

#define EDEADLOCK       EDEADLK

extern int *__errno_location(void);
```

11.3.~~2~~7 fcntl.h

```

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

~~11.3.3 inttypes.h~~

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

~~11.3.4 limits.h~~

~~#define LONG_MAX   0x7FFFFFFFFFFFFFFFL~~
~~#define ULONG_MAX   0xFFFFFFFFFFFFFFFFUL~~

~~#define CHAR_MAX   SCHAR_MAX~~
~~#define CHAR_MIN   SCHAR_MIN~~

~~#define PTHREAD_STACK_MIN   196608~~

~~11.3.5 setjmp.h~~

```

extern int lockf64(int, int, off64_t);
extern int fcntl(int, int, ...);
```

11.3.8 fmtmsg.h

```
extern int fmtmsg(long int, const char *, int, const char *, const char
*,
                  const char *);
```

11.3.9 fnmatch.h

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

11.3.10 ftw.h

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

11.3.11 getopt.h

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

~~11.3.6 signal.h~~

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

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

~~struct sigaction~~
~~{~~
~~ union~~
~~ {~~
~~   sighandler_t _sa_handler;~~
~~   void (*_sa_sigaction) (int, siginfo_t *, void *);~~
~~ }~~
~~ __sigaction_handler;~~
~~ unsigned long int sa_flags;~~
~~ sigset_t sa_mask;~~
~~}~~
~~ ;~~
~~#define MINSIGSTKSZ   131027~~
~~#define SIGSTKSZ   262144~~

~~struct ia64_fpreg~~
~~{~~
~~ union~~
~~ {~~
~~   unsigned long int bits[2];~~
~~   long double __dummy;~~
~~ }~~
~~ u;~~
~~}~~
~~ ;~~

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

11.3.12 glob.h

```

extern int glob(const char *, int,
                int (*__errfunc) (const char *p1, int p2)
                , glob_t *);
extern int glob64(const char *, int,
                  int (*__errfunc) (const char *p1, int p2)
                  , glob64_t *);
extern void globfree(glob_t *);
extern void globfree64(glob64_t *);
```

11.3.13 grp.h

```

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

11.3.14 iconv.h

```

extern size_t iconv(iconv_t, char **, size_t *, char **, size_t *);
extern int iconv_close(iconv_t);
extern iconv_t iconv_open(char *, char *);
```

11.3.15 inttypes.h

```

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

~~struct sigcontext~~
~~{~~
~~  unsigned long int sc_flags;~~
~~  unsigned long int sc_nat;~~
~~  stack_t sc_stack;~~
~~  unsigned long int sc_ip;~~
~~  unsigned long int sc_cfm;~~
~~  unsigned long int sc_um;~~
~~  unsigned long int sc_ar_rsc;~~
~~  unsigned long int sc_ar_bsp;~~
~~  unsigned long int sc_ar_rnat;~~
~~  unsigned long int sc_ar_ccv;~~
~~  unsigned long int sc_ar_unat;~~
~~  unsigned long int sc_ar_fpsr;~~
~~  unsigned long int sc_ar_pfs;~~
~~  unsigned long int sc_ar_lc;~~
~~  unsigned long int sc_pr;~~
~~  unsigned long int sc_br[8];~~
~~  unsigned long int sc_gr[32];~~
~~  struct ia64_fpreg sc_fr[128];~~
~~  unsigned long int sc_rbs_base;~~
~~  unsigned long int sc_loadrs;~~
~~  unsigned long int sc_ar25;~~
~~  unsigned long int sc_ar26;~~
~~  unsigned long int sc_rsvd[12];~~
~~  unsigned long int sc_mask;~~
~~}~~
~~ ;~~
```

~~11.3.7 stddef.h~~

```
~~typedef long int ptrdiff_t;~~
~~typedef unsigned long int size_t;~~
```

~~11.3.8 stdio.h~~

```
~~#define __IO_FILE_SIZE  216~~
```

~~11.3.9 sys/ioctl.h~~

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

~~11.3.10 sys/ipc.h~~

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

~~11.3.11 sys/mman.h~~

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

~~11.3.12 sys/msg.h~~

```
~~struct msqid_ds~~
~~{~~
~~  struct ipc_perm msg_perm;~~
~~  time_t msg_stime;~~
~~  time_t msg_rtime;~~
~~  time_t msg_ctime;~~
~~  unsigned long int __msg_cbytes;~~
~~  unsigned long int msg_qnum;~~
~~    unsigned long int msg_qbytes;~~
~~    pid_t msg_lspid;~~
~~    pid_t msg_lrpid;~~
~~    unsigned long int __unused1;~~
~~    unsigned long int __unused2;~~
~~}~~
~~;~~
```

~~11.3.13 sys/sem.h~~

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

11.3.16 langinfo.h

```

extern char *nl_langinfo(nl_item);
```

11.3.17 libgen.h

```

extern char *basename(const char *);
extern char *dirname(char *);
```

11.3.18 libintl.h

```

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

11.3.19 limits.h

```

#define LONG_MAX        0x7FFFFFFFFFFFFFFFL
#define ULONG_MAX       0xFFFFFFFFFFFFFFFFUL

#define CHAR_MAX        SCHAR_MAX
#define CHAR_MIN        SCHAR_MIN

#define PTHREAD_STACK_MIN       196608
```

11.3.20 locale.h

```

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

~~11.3.14 sys/shm.h~~

```

~~#define SHMLBA  (1024*1024)~~

~~struct shmid_ds~~
~~{~~
~~    struct ipc_perm shm_perm;~~
~~    size_t shm_segsz;~~
~~    time_t shm_atime;~~
~~    time_t shm_dtime;~~
~~    time_t shm_ctime;~~
~~    pid_t shm_cpid;~~
~~    pid_t shm_lpid;~~
~~    unsigned long int shm_nattch;~~
~~    unsigned long int __unused1;~~
~~    unsigned long int __unused2;~~
~~}~~
~~  ;~~
```

~~11.3.15 sys/socket.h~~

```

~~typedef uint64_t __ss_aligntype;~~

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

~~11.3.16 sys/stat.h~~

```

~~#define _STAT_VER       1~~

~~struct stat~~
~~{~~
~~    dev_t st_dev;~~
~~    ino_t st_ino;~~
~~    nlink_t st_nlink;~~
~~    mode_t st_mode;~~
~~    uid_t st_uid;~~
~~    gid_t st_gid;~~
~~    unsigned int pad0;~~
~~    dev_t st_rdev;~~
~~    off_t st_size;~~
~~    struct timespec st_atim;~~
~~    struct timespec st_mtim;~~
~~    struct timespec st_ctim;~~
~~    blksize_t st_blksize;~~
~~    blkcnt_t st_blocks;~~
~~    unsigned long int __unused[3];~~
~~}~~
~~  ;~~
~~struct stat64~~
~~{~~
~~    dev_t st_dev;~~
~~    ino64_t st_ino;~~
~~    nlink_t st_nlink;~~
~~    mode_t st_mode;~~
~~    uid_t st_uid;~~
~~    gid_t st_gid;~~
~~    unsigned int pad0;~~
~~    dev_t st_rdev;~~
~~    off_t st_size;~~
~~    struct timespec st_atim;~~
~~    struct timespec st_mtim;~~
~~    struct timespec st_ctim;~~
~~    blksize_t st_blksize;~~
~~    blkcnt64_t st_blocks;~~
~~    unsigned long int __unused[3];~~
~~}~~
~~ ;~~
```

~~11.3.17 sys/statvfs.h~~

```
extern struct lconv *localeconv(void);
extern char *setlocale(int, const char *);
extern locale_t uselocale(locale_t);
extern void freelocale(locale_t);
extern locale_t duplocale(locale_t);
extern locale_t newlocale(int, const char *, locale_t);
```

11.3.21 monetary.h

```

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

11.3.22 net/if.h

```

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

11.3.23 netdb.h

```

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

11.3.24 netinet/in.h

```

extern int bindresvport(int, struct sockaddr_in *);
```

11.3.25 netinet/ip.h

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

11.3.26 netinet/tcp.h

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

11.3.27 netinet/udp.h

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

~~11.3.18 sys/types.h~~

```
~~typedef long int int64_t;~~

~~typedef int64_t ssize_t;~~

~~#define __FDSET_LONGS    16~~
```

~~11.3.19 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~~
```

~~11.3.20 ucontext.h~~

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

11.3.28 nl_types.h

```

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

11.3.29 poll.h

```

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

11.3.30 pty.h

```

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

11.3.31 pwd.h

```

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

11.3.32 regex.h

```

extern int regcomp(regex_t *, const char *, int);
extern size_t regerror(int, const regex_t *, char *, size_t);
extern int regexec(const regex_t *, const char *, size_t, regmatch_t,
int);
extern void regfree(regex_t *);
```

11.3.33 rpc/auth.h

```

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

11.3.34 rpc/clnt.h

```

extern struct CLIENT *clnt_create(const char *, const u_long, const
u_long,
                                  const char *);
extern void clnt_pcreateerror(const char *);
extern void clnt_perrno(enum clnt_stat);
extern void clnt_perror(struct CLIENT *, const char *);
extern char *clnt_spcreateerror(const char *);
extern char *clnt_sperrno(enum clnt_stat);
extern char *clnt_sperror(struct CLIENT *, const char *);
```

11.3.35 rpc/pmap_clnt.h

```

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

11.3.36 rpc/rpc_msg.h

```

~~#define _SC_GR0_OFFSET (((char *) & ((struct sigcontext *) 0)->sc_gr[0])~~
~~- (char *) 0)~~

~~typedef struct sigcontext mcontext_t;~~

~~typedef struct ucontext~~
~~{~~
~~  union~~
~~  {~~
~~    mcontext_t _mc;~~
~~    struct~~
~~    {~~
~~      unsigned long int _pad[_SC_GR0_OFFSET / 8];~~
~~      struct ucontext *_link;~~
~~    }~~
~~    _uc;~~
~~  }~~
~~  _u;~~
~~}~~
~~ucontext_t;~~
```

~~11.3.21 unistd.h~~

```

~~typedef long int intptr_t;~~
```

~~11.3.22 utmp.h~~

```
extern bool_t xdr_callhdr(XDR *, struct rpc_msg *);
```

11.3.37 rpc/svc.h

```

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

11.3.38 rpc/types.h

```

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

11.3.39 rpc/xdr.h

```

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

11.3.40 sched.h

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

```
extern int sched_get_priority_max(int);
extern int sched_get_priority_min(int);
extern int sched_getparam(pid_t, struct sched_param *);
extern int sched_getscheduler(pid_t);
extern int sched_rr_get_interval(pid_t, struct timespec *);
extern int sched_setparam(pid_t, const struct sched_param *);
extern int sched_setscheduler(pid_t, int, const struct sched_param *);
extern int sched_yield(void);
```

11.3.41 search.h

```
~~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];~~
~~}~~
~~ ;~~
```

~~11.3.23 utmpx.h~~

```
extern int hcreate(size_t);
extern ENTRY *hsearch(ENTRY, ACTION);
extern void insque(void *, void *);
extern void *lfind(const void *, const void *, size_t *, size_t,
                   __compar_fn_t);
extern void *lsearch(const void *, void *, size_t *, size_t,
                     __compar_fn_t);
extern void remque(void *);
extern void hdestroy(void);
extern void *tdelete(const void *, void **, __compar_fn_t);
extern void *tfind(const void *, void *const *, __compar_fn_t);
extern void *tsearch(const void *, void **, __compar_fn_t);
extern void twalk(const void *, __action_fn_t);
```

11.3.42 setjmp.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];~~
~~}~~
~~ ;~~
typedef long int __jmp_buf[70] __attribute__ ((aligned(16)));

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

11.3.43 signal.h

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

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

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

#define MINSIGSTKSZ     131027
#define SIGSTKSZ        262144

struct ia64_fpreg {
    union {
        unsigned long int bits[2];
        long double __dummy;
    } u;
};

struct sigcontext {
    unsigned long int sc_flags;
    unsigned long int sc_nat;
    stack_t sc_stack;
    unsigned long int sc_ip;
    unsigned long int sc_cfm;
    unsigned long int sc_um;
    unsigned long int sc_ar_rsc;
    unsigned long int sc_ar_bsp;
    unsigned long int sc_ar_rnat;
    unsigned long int sc_ar_ccv;
    unsigned long int sc_ar_unat;
    unsigned long int sc_ar_fpsr;
    unsigned long int sc_ar_pfs;
    unsigned long int sc_ar_lc;
    unsigned long int sc_pr;
    unsigned long int sc_br[8];
    unsigned long int sc_gr[32];
    struct ia64_fpreg sc_fr[128];
    unsigned long int sc_rbs_base;
    unsigned long int sc_loadrs;
    unsigned long int sc_ar25;
    unsigned long int sc_ar26;
    unsigned long int sc_rsvd[12];
    unsigned long int sc_mask;
};
extern int __libc_current_sigrtmax(void);
extern int __libc_current_sigrtmin(void);
extern sighandler_t __sysv_signal(int, sighandler_t);
extern char *const _sys_siglist(void);
extern int killpg(pid_t, int);
extern void psignal(int, const char *);
extern int raise(int);
extern int sigaddset(sigset_t *, int);
extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
extern int sigdelset(sigset_t *, int);
extern int sigemptyset(sigset_t *);
extern int sigfillset(sigset_t *);
extern int sighold(int);
extern int sigignore(int);
extern int siginterrupt(int, int);
extern int sigisemptyset(const sigset_t *);
extern int sigismember(const sigset_t *, int);
extern int sigorset(sigset_t *, const sigset_t *, const sigset_t *);
extern int sigpending(sigset_t *);
extern int sigrelse(int);
extern sighandler_t sigset(int, sighandler_t);
extern int pthread_kill(pthread_t, int);
extern int pthread_sigmask(int, sigset_t *, sigset_t *);
extern int sigaction(int, const struct sigaction *, struct sigaction *);
extern int sigwait(sigset_t *, int *);
extern int kill(pid_t, int);
extern int sigaltstack(const struct sigaltstack *, struct sigaltstack
*);
extern sighandler_t signal(int, sighandler_t);
extern int sigpause(int);
extern int sigprocmask(int, const sigset_t *, sigset_t *);
extern int sigreturn(struct sigcontext *);
extern int sigsuspend(const sigset_t *);
extern int sigqueue(pid_t, int, const union sigval);
extern int sigwaitinfo(const sigset_t *, siginfo_t *);
extern int sigtimedwait(const sigset_t *, siginfo_t *,
                        const struct timespec *);
extern sighandler_t bsd_signal(int, sighandler_t);
```

11.3.44 stddef.h

```

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

11.3.45 stdio.h

```

#define __IO_FILE_SIZE  216

extern char *const _sys_errlist(void);
extern void clearerr(FILE *);
extern int fclose(FILE *);
extern FILE *fdopen(int, const char *);
extern int fflush_unlocked(FILE *);
extern int fileno(FILE *);
extern FILE *fopen(const char *, const char *);
extern int fprintf(FILE *, const char *, ...);
extern int fputc(int, FILE *);
extern FILE *freopen(const char *, const char *, FILE *);
extern FILE *freopen64(const char *, const char *, FILE *);
extern int fscanf(FILE *, const char *, ...);
extern int fseek(FILE *, long int, int);
extern int fseeko(FILE *, off_t, int);
extern int fseeko64(FILE *, loff_t, int);
extern off_t ftello(FILE *);
extern loff_t ftello64(FILE *);
extern int getchar(void);
extern int getchar_unlocked(void);
extern int getw(FILE *);
extern int pclose(FILE *);
extern void perror(const char *);
extern FILE *popen(const char *, const char *);
extern int printf(const char *, ...);
extern int putc_unlocked(int, FILE *);
extern int putchar(int);
extern int putchar_unlocked(int);
extern int putw(int, FILE *);
extern int remove(const char *);
extern void rewind(FILE *);
extern int scanf(const char *, ...);
extern void setbuf(FILE *, char *);
extern int sprintf(char *, const char *, ...);
extern int sscanf(const char *, const char *, ...);
extern FILE *stderr(void);
extern FILE *stdin(void);
extern FILE *stdout(void);
extern char *tempnam(const char *, const char *);
extern FILE *tmpfile64(void);
extern FILE *tmpfile(void);
extern char *tmpnam(char *);
extern int vfprintf(FILE *, const char *, va_list);
extern int vprintf(const char *, va_list);
extern int feof(FILE *);
extern int ferror(FILE *);
extern int fflush(FILE *);
extern int fgetc(FILE *);
extern int fgetpos(FILE *, fpos_t *);
extern char *fgets(char *, int, FILE *);
extern int fputs(const char *, FILE *);
extern size_t fread(void *, size_t, size_t, FILE *);
extern int fsetpos(FILE *, const fpos_t *);
extern long int ftell(FILE *);
extern size_t fwrite(const void *, size_t, size_t, FILE *);
extern int getc(FILE *);
extern int putc(int, FILE *);
extern int puts(const char *);
extern int setvbuf(FILE *, char *, int, size_t);
extern int snprintf(char *, size_t, const char *, ...);
extern int ungetc(int, FILE *);
extern int vsnprintf(char *, size_t, const char *, va_list);
extern int vsprintf(char *, const char *, va_list);
extern void flockfile(FILE *);
extern int asprintf(char **, const char *, ...);
extern int fgetpos64(FILE *, fpos64_t *);
extern FILE *fopen64(const char *, const char *);
extern int fsetpos64(FILE *, const fpos64_t *);
extern int ftrylockfile(FILE *);
extern void funlockfile(FILE *);
extern int getc_unlocked(FILE *);
extern void setbuffer(FILE *, char *, size_t);
extern int vasprintf(char **, const char *, va_list);
extern int vdprintf(int, const char *, va_list);
extern int vfscanf(FILE *, const char *, va_list);
extern int vscanf(const char *, va_list);
extern int vsscanf(const char *, const char *, va_list);
extern size_t __fpending(FILE *);
```

11.3.46 stdlib.h

```

extern double __strtod_internal(const char *, char **, int);
extern float __strtof_internal(const char *, char **, int);
extern long int __strtol_internal(const char *, char **, int, int);
extern long double __strtold_internal(const char *, char **, int);
extern long long int __strtoll_internal(const char *, char **, int, int);
extern unsigned long int __strtoul_internal(const char *, char **, int,
                                            int);
extern unsigned long long int __strtoull_internal(const char *, char **,
                                                  int, int);
extern long int a64l(const char *);
extern void abort(void);
extern int abs(int);
```

```
        extern double atof(const char *);
        extern int atoi(char *);
        extern long int atol(char *);
        extern long long int atoll(const char *);
        extern void *bsearch(const void *, const void *, size_t, size_t,
                             __compar_fn_t);
        extern div_t div(int, int);
        extern double drand48(void);
        extern char *ecvt(double, int, int *, int *);
        extern double erand48(unsigned short);
        extern void exit(int);
        extern char *fcvt(double, int, int *, int *);
        extern char *gcvt(double, int, char *);
        extern char *getenv(const char *);
        extern int getsubopt(char **, char *const *, char **);
        extern int grantpt(int);
        extern long int jrand48(unsigned short);
        extern char *l64a(long int);
        extern long int labs(long int);
        extern void lcong48(unsigned short);
        extern ldiv_t ldiv(long int, long int);
        extern long long int llabs(long long int);
        extern lldiv_t lldiv(long long int, long long int);
        extern long int lrand48(void);
        extern int mblen(const char *, size_t);
        extern size_t mbstowcs(wchar_t *, const char *, size_t);
        extern int mbtowc(wchar_t *, const char *, size_t);
        extern char *mktemp(char *);
        extern long int mrand48(void);
        extern long int nrand48(unsigned short);
        extern char *ptsname(int);
        extern int putenv(char *);
        extern void qsort(void *, size_t, size_t, __compar_fn_t);
        extern int rand(void);
        extern int rand_r(unsigned int *);
        extern unsigned short *seed48(unsigned short);
        extern void srand48(long int);
        extern int unlockpt(int);
        extern size_t wcstombs(char *, const wchar_t *, size_t);
        extern int wctomb(char *, wchar_t);
        extern int system(const char *);
        extern void *calloc(size_t, size_t);
        extern void free(void *);
        extern char *initstate(unsigned int, char *, size_t);
        extern void *malloc(size_t);
        extern long int random(void);
        extern void *realloc(void *, size_t);
        extern char *setstate(char *);
        extern void srand(unsigned int);
        extern void srandom(unsigned int);
        extern double strtod(char *, char **);
        extern float strtof(const char *, char **);
        extern long int strtol(char *, char **, int);
        extern long double strtold(const char *, char **);
        extern long long int strtoll(const char *, char **, int);
        extern long long int strtoq(const char *, char **, int);
        extern unsigned long int strtoul(const char *, char **, int);
        extern unsigned long long int strtoull(const char *, char **, int);
        extern unsigned long long int strtouq(const char *, char **, int);
        extern void _Exit(int);
        extern size_t __ctype_get_mb_cur_max(void);
        extern char **environ(void);
        extern char *realpath(const char *, char *);
        extern int setenv(const char *, const char *, int);
```

```
extern int unsetenv(const char *);
extern int getloadavg(double, int);
extern int mkstemp64(char *);
extern int posix_memalign(void **, size_t, size_t);
extern int posix_openpt(int);
```

11.3.47 string.h

```

extern void *__mempcpy(void *, const void *, size_t);
extern char *__stpcpy(char *, const char *);
extern char *__strtok_r(char *, const char *, char **);
extern void bcopy(void *, void *, size_t);
extern void *memchr(void *, int, size_t);
extern int memcmp(void *, void *, size_t);
extern void *memcpy(void *, void *, size_t);
extern void *memmem(const void *, size_t, const void *, size_t);
extern void *memmove(void *, const void *, size_t);
extern void *memset(void *, int, size_t);
extern char *strcat(char *, const char *);
extern char *strchr(char *, int);
extern int strcmp(char *, char *);
extern int strcoll(const char *, const char *);
extern char *strcpy(char *, char *);
extern size_t strcspn(const char *, const char *);
extern char *strerror(int);
extern size_t strlen(char *);
extern char *strncat(char *, char *, size_t);
extern int strncmp(char *, char *, size_t);
extern char *strncpy(char *, char *, size_t);
extern char *strpbrk(const char *, const char *);
extern char *strrchr(char *, int);
extern char *strsignal(int);
extern size_t strspn(const char *, const char *);
extern char *strstr(char *, char *);
extern char *strtok(char *, const char *);
extern size_t strxfrm(char *, const char *, size_t);
extern int bcmp(void *, void *, size_t);
extern void bzero(void *, size_t);
extern int ffs(int);
extern char *index(char *, int);
extern void *memccpy(void *, const void *, int, size_t);
extern char *rindex(char *, int);
extern int strcasecmp(char *, char *);
extern char *strdup(char *);
extern int strncasecmp(char *, char *, size_t);
extern char *strndup(const char *, size_t);
extern size_t strnlen(const char *, size_t);
extern char *strsep(char **, const char *);
extern char *strerror_r(int, char *, size_t);
extern char *strtok_r(char *, const char *, char **);
extern char *strcasestr(const char *, const char *);
extern char *stpcpy(char *, const char *);
extern char *stpncpy(char *, const char *, size_t);
extern void *memrchr(const void *, int, size_t);
```

11.3.48 sys/file.h

```

extern int flock(int, int);
```

11.3.49 sys/ioctl.h

```

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

extern int ioctl(int, unsigned long int, ...);
```

11.3.50 sys/ipc.h

```

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

extern key_t ftok(char *, int);
```

11.3.51 sys/mman.h

```

#define MCL_CURRENT     1
#define MCL_FUTURE      2

extern int msync(void *, size_t, int);
extern int mlock(const void *, size_t);
extern int mlockall(int);
extern void *mmap(void *, size_t, int, int, int, off_t);
extern int mprotect(void *, size_t, int);
extern int munlock(const void *, size_t);
extern int munlockall(void);
extern int munmap(void *, size_t);
extern void *mmap64(void *, size_t, int, int, int, off64_t);
extern int shm_open(const char *, int, mode_t);
extern int shm_unlink(const char *);
```

11.3.52 sys/msg.h

```

struct msqid_ds {
    struct ipc_perm msg_perm;
    time_t msg_stime;
    time_t msg_rtime;
    time_t msg_ctime;
    unsigned long int __msg_cbytes;
    unsigned long int msg_qnum;
    unsigned long int msg_qbytes;
    pid_t msg_lspid;
    pid_t msg_lrpid;
    unsigned long int __unused1;
    unsigned long int __unused2;
};
extern int msgctl(int, int, struct msqid_ds *);
extern int msgget(key_t, int);
extern int msgrcv(int, void *, size_t, long int, int);
extern int msgsnd(int, const void *, size_t, int);
```

11.3.53 sys/param.h

```

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

11.3.54 sys/poll.h

```

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

11.3.55 sys/resource.h

```

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

11.3.56 sys/sem.h

```

struct semid_ds {
    struct ipc_perm sem_perm;
    time_t sem_otime;
    time_t sem_ctime;
    unsigned long int sem_nsems;
    unsigned long int __unused1;
    unsigned long int __unused2;
};
extern int semctl(int, int, int, ...);
extern int semget(key_t, int, int);
extern int semop(int, struct sembuf *, size_t);
```

11.3.57 sys/shm.h

```

#define SHMLBA  (1024*1024)

struct shmid_ds {
    struct ipc_perm shm_perm;
    size_t shm_segsz;
    time_t shm_atime;
    time_t shm_dtime;
    time_t shm_ctime;
    pid_t shm_cpid;
    pid_t shm_lpid;
    unsigned long int shm_nattch;
    unsigned long int __unused1;
    unsigned long int __unused2;
};
extern int __getpagesize(void);
extern void *shmat(int, const void *, int);
extern int shmctl(int, int, struct shmid_ds *);
extern int shmdt(const void *);
extern int shmget(key_t, size_t, int);
```

11.3.58 sys/socket.h

```

typedef uint64_t __ss_aligntype;

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

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

11.3.59 sys/stat.h

```

#define _STAT_VER       1

struct stat {
    dev_t st_dev;
    ino_t st_ino;
    nlink_t st_nlink;
    mode_t st_mode;
    uid_t st_uid;
    gid_t st_gid;
    unsigned int pad0;
    dev_t st_rdev;
    off_t st_size;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    blksize_t st_blksize;
    blkcnt_t st_blocks;
    unsigned long int __unused[3];
};
struct stat64 {
    dev_t st_dev;
    ino64_t st_ino;
    nlink_t st_nlink;
```

```
        mode_t st_mode;
        uid_t st_uid;
        gid_t st_gid;
        unsigned int pad0;
        dev_t st_rdev;
        off_t st_size;
        struct timespec st_atim;
        struct timespec st_mtim;
        struct timespec st_ctim;
        blksize_t st_blksize;
        blkcnt64_t st_blocks;
        unsigned long int __unused[3];
    };

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

11.3.60 sys/statvfs.h

```

    struct statvfs {
        unsigned long int f_bsize;
        unsigned long int f_frsize;
        fsblkcnt64_t f_blocks;
        fsblkcnt64_t f_bfree;
        fsblkcnt64_t f_bavail;
        fsfilcnt64_t f_files;
        fsfilcnt64_t f_ffree;
        fsfilcnt64_t f_favail;
        unsigned long int f_fsid;
        unsigned long int f_flag;
        unsigned long int f_namemax;
        unsigned int __f_spare[6];
    };
    struct statvfs64 {
        unsigned long int f_bsize;
        unsigned long int f_frsize;
        fsblkcnt64_t f_blocks;
        fsblkcnt64_t f_bfree;
        fsblkcnt64_t f_bavail;
        fsfilcnt64_t f_files;
        fsfilcnt64_t f_ffree;
        fsfilcnt64_t f_favail;
        unsigned long int f_fsid;
        unsigned long int f_flag;
        unsigned long int f_namemax;
        unsigned int __f_spare[6];
    };
    extern int fstatvfs(int, struct statvfs *);
    extern int fstatvfs64(int, struct statvfs64 *);
    extern int statvfs(const char *, struct statvfs *);
    extern int statvfs64(const char *, struct statvfs64 *);
```

11.3.61 sys/time.h

```
extern int getitimer(__itimer_which_t, struct itimerval *);
extern int setitimer(__itimer_which_t, const struct itimerval *,
                     struct itimerval *);
extern int adjtime(const struct timeval *, struct timeval *);
extern int gettimeofday(struct timeval *, struct timezone *);
extern int utimes(const char *, const struct timeval *);
```

11.3.62 sys/timeb.h

```
extern int ftime(struct timeb *);
```

11.3.63 sys/times.h

```
extern clock_t times(struct tms *);
```

11.3.64 sys/types.h

```
typedef long int int64_t;

typedef int64_t ssize_t;

#define __FDSET_LONGS   16
```

11.3.65 sys/uio.h

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

11.3.66 sys/un.h

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

11.3.67 sys/utsname.h

```
extern int uname(struct utsname *);
```

11.3.68 sys/wait.h

```
extern pid_t wait(int *);
extern pid_t waitpid(pid_t, int *, int);
extern pid_t wait4(pid_t, int *, int, struct rusage *);
```

11.3.69 syslog.h

```
extern void closelog(void);
extern void openlog(const char *, int, int);
extern int setlogmask(int);
extern void syslog(int, const char *, ...);
extern void vsyslog(int, const char *, va_list);
```

11.3.70 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

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

11.3.71 time.h

```

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

11.3.72 ucontext.h

```

#define _SC_GR0_OFFSET  \
        (((char *) &((struct sigcontext *) 0)->sc_gr[0]) - (char *) 0)

typedef struct sigcontext mcontext_t;

typedef struct ucontext {
    union {
        mcontext_t _mc;
        struct {
            unsigned long int _pad[_SC_GR0_OFFSET / 8];
            struct ucontext *_link;
        } _uc;
    } _u;
} ucontext_t;
extern int getcontext(ucontext_t *);
extern int makecontext(ucontext_t *, void (*func) (void)
                       , int, ...);
extern int setcontext(const struct ucontext *);
extern int swapcontext(ucontext_t *, const struct ucontext *);
```

11.3.73 ulimit.h

```

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

11.3.74 unistd.h

```

typedef long int intptr_t;

extern char **__environ(void);
extern pid_t __getpgid(pid_t);
extern void _exit(int);
extern int acct(const char *);
extern unsigned int alarm(unsigned int);
extern int chown(const char *, uid_t, gid_t);
extern int chroot(const char *);
extern size_t confstr(int, char *, size_t);
extern int creat(const char *, mode_t);
extern int creat64(const char *, mode_t);
extern char *ctermid(char *);
extern char *cuserid(char *);
extern int daemon(int, int);
extern int execl(const char *, const char *, ...);
extern int execle(const char *, const char *, ...);
extern int execlp(const char *, const char *, ...);
extern int execv(const char *, char *const);
extern int execvp(const char *, char *const);
extern int fdatasync(int);
extern int ftruncate64(int, off64_t);
extern long int gethostid(void);
extern char *getlogin(void);
extern int getlogin_r(char *, size_t);
extern int getopt(int, char *const, const char *);
extern pid_t getpgrp(void);
extern pid_t getsid(pid_t);
extern char *getwd(char *);
extern int lockf(int, int, off_t);
extern int mkstemp(char *);
extern int nice(int);
extern char *optarg(void);
extern int opterr(void);
extern int optind(void);
extern int optopt(void);
extern int rename(const char *, const char *);
extern int setegid(gid_t);
extern int seteuid(uid_t);
extern int sethostname(const char *, size_t);
```

```
        extern int setpgrp(void);
        extern void swab(const void *, void *, ssize_t);
        extern void sync(void);
        extern pid_t tcgetpgrp(int);
        extern int tcsetpgrp(int, pid_t);
        extern int truncate(const char *, off_t);
        extern int truncate64(const char *, off64_t);
        extern char *ttyname(int);
        extern unsigned int ualarm(useconds_t, useconds_t);
        extern int usleep(useconds_t);
        extern int close(int);
        extern int fsync(int);
        extern off_t lseek(int, off_t, int);
        extern int open(const char *, int, ...);
        extern int pause(void);
        extern ssize_t read(int, void *, size_t);
        extern ssize_t write(int, const void *, size_t);
        extern char *crypt(char *, char *);
        extern void encrypt(char *, int);
        extern void setkey(const char *);
        extern int access(const char *, int);
        extern int brk(void *);
        extern int chdir(const char *);
        extern int dup(int);
        extern int dup2(int, int);
        extern int execve(const char *, char *const, char *const);
        extern int fchdir(int);
        extern int fchown(int, uid_t, gid_t);
        extern pid_t fork(void);
        extern gid_t getegid(void);
        extern uid_t geteuid(void);
        extern gid_t getgid(void);
        extern int getgroups(int, gid_t);
        extern int gethostname(char *, size_t);
        extern pid_t getpgid(pid_t);
        extern pid_t getpid(void);
        extern uid_t getuid(void);
        extern int lchown(const char *, uid_t, gid_t);
        extern int link(const char *, const char *);
        extern int mkdir(const char *, mode_t);
        extern long int pathconf(const char *, int);
        extern int pipe(int);
        extern int readlink(const char *, char *, size_t);
        extern int rmdir(const char *);
        extern void *sbrk(ptrdiff_t);
        extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
        extern int setgid(gid_t);
        extern int setpgid(pid_t, pid_t);
        extern int setregid(gid_t, gid_t);
        extern int setreuid(uid_t, uid_t);
        extern pid_t setsid(void);
        extern int setuid(uid_t);
        extern unsigned int sleep(unsigned int);
        extern int symlink(const char *, const char *);
        extern long int sysconf(int);
        extern int unlink(const char *);
        extern pid_t vfork(void);
        extern ssize_t pread(int, void *, size_t, off_t);
        extern ssize_t pwrite(int, const void *, size_t, off_t);
        extern char **_environ(void);
        extern long int fpathconf(int, int);
        extern int ftruncate(int, off_t);
        extern char *getcwd(char *, size_t);
        extern int getpagesize(void);
```

```
extern pid_t getppid(void);
extern int isatty(int);
extern loff_t lseek64(int, loff_t, int);
extern int open64(const char *, int, ...);
extern ssize_t pread64(int, void *, size_t, off64_t);
extern ssize_t pwrite64(int, const void *, size_t, off64_t);
extern int ttyname_r(int, char *, size_t);
```

11.3.75 utime.h

```

extern int utime(const char *, const struct utimbuf *);
```

11.3.76 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];
};

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

11.3.77 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];
};

```

```
extern void endutxent(void);
extern struct utmpx *getutxent(void);
extern struct utmpx *getutxid(const struct utmpx *);
extern struct utmpx *getutxline(const struct utmpx *);
extern struct utmpx *pututxline(const struct utmpx *);
extern void setutxent(void);
```

11.3.78 wchar.h

```

extern double __wcstod_internal(const wchar_t *, wchar_t * *, int);
extern float __wcstof_internal(const wchar_t *, wchar_t * *, int);
extern long int __wcstol_internal(const wchar_t *, wchar_t * *, int,
int);
extern long double __wcstold_internal(const wchar_t *, wchar_t * *, int);
extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t *
*,
                                            int, int);
extern wchar_t *wcscat(wchar_t *, const wchar_t *);
extern wchar_t *wcschr(const wchar_t *, wchar_t);
extern int wcscmp(const wchar_t *, const wchar_t *);
extern int wcscoll(const wchar_t *, const wchar_t *);
extern wchar_t *wcscpy(wchar_t *, const wchar_t *);
extern size_t wcscspn(const wchar_t *, const wchar_t *);
extern wchar_t *wcsdup(const wchar_t *);
extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wcspbrk(const wchar_t *, const wchar_t *);
extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
extern size_t wcsspn(const wchar_t *, const wchar_t *);
extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t * *);
extern int wcswidth(const wchar_t *, size_t);
extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
extern int wctob(wint_t);
extern int wcwidth(wchar_t);
extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
extern size_t mbrlen(const char *, size_t, mbstate_t *);
extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
extern int mbsinit(const mbstate_t *);
extern size_t mbsnrtowcs(wchar_t *, const char **, size_t, size_t,
                         mbstate_t *);
extern size_t mbsrtowcs(wchar_t *, const char **, size_t, mbstate_t *);
extern wchar_t *wcpcpy(wchar_t *, const wchar_t *);
extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
extern size_t wcrtomb(char *, wchar_t, mbstate_t *);
extern size_t wcslen(const wchar_t *);
extern size_t wcsnrtombs(char *, const wchar_t * *, size_t, size_t,
                         mbstate_t *);
extern size_t wcsrtombs(char *, const wchar_t * *, size_t, mbstate_t *);
extern double wcstod(const wchar_t *, wchar_t * *);
extern float wcstof(const wchar_t *, wchar_t * *);
extern long int wcstol(const wchar_t *, wchar_t * *, int);
extern long double wcstold(const wchar_t *, wchar_t * *);
extern long long int wcstoq(const wchar_t *, wchar_t * *, int);
extern unsigned long int wcstoul(const wchar_t *, wchar_t * *, int);
extern unsigned long long int wcstouq(const wchar_t *, wchar_t * *, int);
extern wchar_t *wcswcs(const wchar_t *, const wchar_t *);
extern int wcscasecmp(const wchar_t *, const wchar_t *);
extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
extern size_t wcsnlen(const wchar_t *, size_t);
extern long long int wcstoll(const wchar_t *, wchar_t * *, int);
extern unsigned long long int wcstoull(const wchar_t *, wchar_t * *, int);
extern wint_t btowc(int);
extern wint_t fgetwc(FILE *);
extern wint_t fgetwc_unlocked(FILE *);
extern wchar_t *fgetws(wchar_t *, int, FILE *);
extern wint_t fputwc(wchar_t, FILE *);
extern int fputws(const wchar_t *, FILE *);
extern int fwide(FILE *, int);
extern int fwprintf(FILE *, const wchar_t *, ...);
extern int fwscanf(FILE *, const wchar_t *, ...);
extern wint_t getwc(FILE *);
extern wint_t getwchar(void);
extern wint_t putwc(wchar_t, FILE *);
extern wint_t putwchar(wchar_t);
extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
extern int swscanf(const wchar_t *, const wchar_t *, ...);
extern wint_t ungetwc(wint_t, FILE *);
extern int vfwprintf(FILE *, const wchar_t *, va_list);
extern int vfwscanf(FILE *, const wchar_t *, va_list);
extern int vswprintf(wchar_t *, size_t, const wchar_t *, va_list);
extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
extern int vwprintf(const wchar_t *, va_list);
extern int vwscanf(const wchar_t *, va_list);
extern size_t wcsftime(wchar_t *, size_t, const wchar_t *,
                       const struct tm *);
extern int wprintf(const wchar_t *, ...);
extern int wscanf(const wchar_t *, ...);
```

11.3.79 wctype.h

```

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

11.3.80 wordexp.h

```

extern int wordexp(const char *, wordexp_t *, int);
extern void wordfree(wordexp_t *);
```

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.6.1

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

[ISOC99] ISO C (1999)
[LSB] ~~this specification~~This Specification
[SUSv2] SUSv2
[SUSv3] ISO POSIX (2003)

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_2.2) [1]~~	~~ccoshl(GLIBC_2.2) [2]~~	~~exp(GLIBC_2.2) [2]~~	~~j1l(GLIBC_2.2) [1]~~	~~powl(GLIBC_2.2) [2]~~
~~__finitef(GLIBC_2.2) [1]~~	~~ccosl(GLIBC_2.2) [2]~~	~~exp2(GLIBC_2.2) [2]~~	~~jn(GLIBC_2.2) [2]~~	~~remainder(GLIBC_2.2) [2]~~
~~__finitel(GLIBC_2.2) [1]~~	~~ceil(GLIBC_2.2) [2]~~	~~exp2f(GLIBC_2.2) [2]~~	~~jnf(GLIBC_2.2) [1]~~	~~remainderf(GLIBC_2.2) [2]~~
~~__fpclassify(GLIBC_2.2) [3]~~	~~ceilf(GLIBC_2.2) [2]~~	~~exp2l(GLIBC_2.2) [2]~~	~~jnl(GLIBC_2.2) [1]~~	~~remainderl(GLIBC_2.2) [2]~~
~~__fpclassifyf(GLIBC_2.2) [3]~~	~~ceill(GLIBC_2.2) [2]~~	~~expf(GLIBC_2.2) [2]~~	~~ldexp(GLIBC_2.2) [2]~~	~~remquo(GLIBC_2.2) [2]~~
~~__fpclassifyl(GLIBC_2.2) [1]~~	~~cexp(GLIBC_2.2) [2]~~	~~expl(GLIBC_2.2) [2]~~	~~ldexpf(GLIBC_2.2) [2]~~	~~remquof(GLIBC_2.2) [2]~~
~~__signbit(GLIBC_2.2) [1]~~	~~cexpf(GLIBC_2.2) [2]~~	~~expm1(GLIBC_2.2) [2]~~	~~ldexpl(GLIBC_2.2) [2]~~	~~remquol(GLIBC_2.2) [2]~~
~~__signbitf(GLIBC_2.2) [1]~~	~~cexpl(GLIBC_2.2) [2]~~	~~expm1f(GLIBC_2.2) [2]~~	~~lgamma(GLIBC_2.2) [2]~~	~~rint(GLIBC_2.2) [2]~~
~~__signbitl(GLIBC_2.2) [1]~~	~~cimag(GLIBC_2.2) [2]~~	~~expm1l(GLIBC_2.2) [2]~~	~~lgamma_r(GLIBC_2.2) [1]~~	~~rintf(GLIBC_2.2) [2]~~
~~acos(GLIBC_2.2) [2]~~	~~cimagf(GLIBC_2.2) [2]~~	~~fabs(GLIBC_2.2) [2]~~	~~lgammaf(GLIBC_2.2) [2]~~	~~rintl(GLIBC_2.2) [2]~~
~~acosf(GLIBC_2.2) [2]~~	~~cimagl(GLIBC_2.2) [2]~~	~~fabsf(GLIBC_2.2) [2]~~	~~lgammaf_r(GLIBC_2.2) [1]~~	~~round(GLIBC_2.2) [2]~~
~~acosh(GLIBC~~	~~clog(GLIBC_2~~	~~fabsl(GLIBC_~~	~~lgammal(GLI~~	~~roundf(GLIB~~

~~_2.2) [2]~~	~~.2) [2]~~	~~2.2) [2]~~	~~BC_2.2) [2]~~	~~C_2.2) [2]~~
~~acoshf(GLIBC_2.2) [2]~~	~~clog10(GLIBC_2.2) [1]~~	~~fdim(GLIBC_2.2) [2]~~	~~lgammal_r(GLIBC_2.2) [1]~~	~~roundl(GLIBC_2.2) [2]~~
~~acoshl(GLIBC_2.2) [2]~~	~~clog10f(GLIBC_2.2) [1]~~	~~fdimf(GLIBC_2.2) [2]~~	~~llrint(GLIBC_2.2) [2]~~	~~scalb(GLIBC_2.2) [2]~~
~~acosl(GLIBC_2.2) [2]~~	~~clog10l(GLIBC_2.2) [1]~~	~~fdiml(GLIBC_2.2) [2]~~	~~llrintf(GLIBC_2.2) [2]~~	~~scalbf(GLIBC_2.2) [1]~~
~~asin(GLIBC_2.2) [2]~~	~~clogf(GLIBC_2.2) [2]~~	~~feclearexcept(GLIBC_2.2) [2]~~	~~llrintl(GLIBC_2.2) [2]~~	~~scalbl(GLIBC_2.2) [1]~~
~~asinf(GLIBC_2.2) [2]~~	~~clogl(GLIBC_2.2) [2]~~	~~fegetenv(GLIBC_2.2) [2]~~	~~llround(GLIBC_2.2) [2]~~	~~scalbln(GLIBC_2.2) [2]~~
~~asinh(GLIBC_2.2) [2]~~	~~conj(GLIBC_2.2) [2]~~	~~fegetexceptflag(GLIBC_2.2) [2]~~	~~llroundf(GLIBC_2.2) [2]~~	~~scalblnf(GLIBC_2.2) [2]~~
~~asinhf(GLIBC_2.2) [2]~~	~~conjf(GLIBC_2.2) [2]~~	~~fegetround(GLIBC_2.2) [2]~~	~~llroundl(GLIBC_2.2) [2]~~	~~scalblnl(GLIBC_2.2) [2]~~
~~asinhl(GLIBC_2.2) [2]~~	~~conjl(GLIBC_2.2) [2]~~	~~feholdexcept(GLIBC_2.2) [2]~~	~~log(GLIBC_2.2) [2]~~	~~scalbn(GLIBC_2.2) [2]~~
~~asinl(GLIBC_2.2) [2]~~	~~copysign(GLIBC_2.2) [2]~~	~~feraiseexcept(GLIBC_2.2) [2]~~	~~log10(GLIBC_2.2) [2]~~	~~scalbnf(GLIBC_2.2) [2]~~
~~atan(GLIBC_2.2) [2]~~	~~copysignf(GLIBC_2.2) [2]~~	~~fesetenv(GLIBC_2.2) [2]~~	~~log10f(GLIBC_2.2) [2]~~	~~scalbnl(GLIBC_2.2) [2]~~
~~atan2(GLIBC_2.2) [2]~~	~~copysignl(GLIBC_2.2) [2]~~	~~fesetexceptflag(GLIBC_2.2) [2]~~	~~log10l(GLIBC_2.2) [2]~~	~~significand(GLIBC_2.2) [1]~~
~~atan2f(GLIBC_2.2) [2]~~	~~cos(GLIBC_2.2) [2]~~	~~fesetround(GLIBC_2.2) [2]~~	~~log1p(GLIBC_2.2) [2]~~	~~significandf(GLIBC_2.2) [1]~~
~~atan2l(GLIBC_2.2) [2]~~	~~cosf(GLIBC_2.2) [2]~~	~~fetestexcept(GLIBC_2.2) [2]~~	~~log1pf(GLIBC_2.2) [2]~~	~~significandl(GLIBC_2.2) [1]~~
~~atanf(GLIBC_2.2) [2]~~	~~cosh(GLIBC_2.2) [2]~~	~~feupdateenv(GLIBC_2.2) [2]~~	~~log1pl(GLIBC_2.2) [2]~~	~~sin(GLIBC_2.2) [2]~~
~~atanh(GLIBC_2.2) [2]~~	~~coshf(GLIBC_2.2) [2]~~	~~finite(GLIBC_2.2) [4]~~	~~log2(GLIBC_2.2) [2]~~	~~sincos(GLIBC_2.2) [1]~~
~~atanhf(GLIBC_2.2) [2]~~	~~coshl(GLIBC_2.2) [2]~~	~~finitef(GLIBC_2.2) [1]~~	~~log2f(GLIBC_2.2) [2]~~	~~sincosf(GLIBC_2.2) [1]~~
~~atanhl(GLIBC_2.2) [2]~~	~~cosl(GLIBC_2.2) [2]~~	~~finitel(GLIBC_2.2) [1]~~	~~log2l(GLIBC_2.2) [2]~~	~~sincosl(GLIBC_2.2) [1]~~
~~atanl(GLIBC_~~	~~cpow(GLIBC_~~	~~floor(GLIBC_~~	~~logb(GLIBC_2~~	~~sinf(GLIBC_2.~~

~~2.2) [2]~~	~~2.2) [2]~~	~~2.2) [2]~~	~~.2) [2]~~	~~2) [2]~~
~~cabs(GLIBC_2.2) [2]~~	~~cpowf(GLIBC_2.2) [2]~~	~~floorf(GLIBC_2.2) [2]~~	~~logbf(GLIBC_2.2) [2]~~	~~sinh(GLIBC_2.2) [2]~~
~~cabsf(GLIBC_2.2) [2]~~	~~cpowl(GLIBC_2.2) [2]~~	~~floorl(GLIBC_2.2) [2]~~	~~logbl(GLIBC_2.2) [2]~~	~~sinhf(GLIBC_2.2) [2]~~
~~cabsl(GLIBC_2.2) [2]~~	~~cproj(GLIBC_2.2) [2]~~	~~fma(GLIBC_2.2) [2]~~	~~logf(GLIBC_2.2) [2]~~	~~sinhl(GLIBC_2.2) [2]~~
~~cacos(GLIBC_2.2) [2]~~	~~cprojf(GLIBC_2.2) [2]~~	~~fmaf(GLIBC_2.2) [2]~~	~~logl(GLIBC_2.2) [2]~~	~~sinl(GLIBC_2.2) [2]~~
~~cacosf(GLIBC_2.2) [2]~~	~~cprojl(GLIBC_2.2) [2]~~	~~fmal(GLIBC_2.2) [2]~~	~~lrint(GLIBC_2.2) [2]~~	~~sqrt(GLIBC_2.2) [2]~~
~~cacosh(GLIBC_2.2) [2]~~	~~creal(GLIBC_2.2) [2]~~	~~fmax(GLIBC_2.2) [2]~~	~~lrintf(GLIBC_2.2) [2]~~	~~sqrtf(GLIBC_2.2) [2]~~
~~cacoshf(GLIBC_2.2) [2]~~	~~crealf(GLIBC_2.2) [2]~~	~~fmaxf(GLIBC_2.2) [2]~~	~~lrintl(GLIBC_2.2) [2]~~	~~sqrtl(GLIBC_2.2) [2]~~
~~cacoshl(GLIBC_2.2) [2]~~	~~creall(GLIBC_2.2) [2]~~	~~fmaxl(GLIBC_2.2) [2]~~	~~lround(GLIBC_2.2) [2]~~	~~tan(GLIBC_2.2) [2]~~
~~cacosl(GLIBC_2.2) [2]~~	~~csin(GLIBC_2.2) [2]~~	~~fmin(GLIBC_2.2) [2]~~	~~lroundf(GLIBC_2.2) [2]~~	~~tanf(GLIBC_2.2) [2]~~
~~carg(GLIBC_2.2) [2]~~	~~csinf(GLIBC_2.2) [2]~~	~~fminf(GLIBC_2.2) [2]~~	~~lroundl(GLIBC_2.2) [2]~~	~~tanh(GLIBC_2.2) [2]~~
~~cargf(GLIBC_2.2) [2]~~	~~csinh(GLIBC_2.2) [2]~~	~~fminl(GLIBC_2.2) [2]~~	~~matherr(GLIBC_2.2) [1]~~	~~tanhf(GLIBC_2.2) [2]~~
~~cargl(GLIBC_2.2) [2]~~	~~csinhf(GLIBC_2.2) [2]~~	~~fmod(GLIBC_2.2) [2]~~	~~modf(GLIBC_2.2) [2]~~	~~tanhl(GLIBC_2.2) [2]~~
~~casin(GLIBC_2.2) [2]~~	~~csinhl(GLIBC_2.2) [2]~~	~~fmodf(GLIBC_2.2) [2]~~	~~modff(GLIBC_2.2) [2]~~	~~tanl(GLIBC_2.2) [2]~~
~~casinf(GLIBC_2.2) [2]~~	~~csinl(GLIBC_2.2) [2]~~	~~fmodl(GLIBC_2.2) [2]~~	~~modfl(GLIBC_2.2) [2]~~	~~tgamma(GLIBC_2.2) [2]~~
~~casinh(GLIBC_2.2) [2]~~	~~csqrt(GLIBC_2.2) [2]~~	~~frexp(GLIBC_2.2) [2]~~	~~nan(GLIBC_2.2) [2]~~	~~tgammaf(GLIBC_2.2) [2]~~
~~casinhf(GLIBC_2.2) [2]~~	~~csqrtf(GLIBC_2.2) [2]~~	~~frexpf(GLIBC_2.2) [2]~~	~~nanf(GLIBC_2.2) [2]~~	~~tgammal(GLIBC_2.2) [2]~~
~~casinhl(GLIBC_2.2) [2]~~	~~csqrtl(GLIBC_2.2) [2]~~	~~frexpl(GLIBC_2.2) [2]~~	~~nanl(GLIBC_2.2) [2]~~	~~trunc(GLIBC_2.2) [2]~~
~~casinl(GLIBC_2.2) [2]~~	~~ctan(GLIBC_2.2) [2]~~	~~gamma(GLIBC_2.2) [4]~~	~~nearbyint(GLIBC_2.2) [2]~~	~~truncf(GLIBC_2.2) [2]~~
~~catan(GLIBC_2.2) [2]~~	~~ctanf(GLIBC_2.2) [2]~~	~~gammaf(GLIBC_2.2) [1]~~	~~nearbyintf(GLIBC_2.2) [2]~~	~~truncl(GLIBC_2.2) [2]~~
~~catanf(GLIBC_2.2) [2]~~	~~ctanh(GLIBC_2.2) [2]~~	~~gammal(GLIBC_2.2) [1]~~	~~nearbyintl(GLIBC_2.2) [2]~~	~~y0(GLIBC_2.2) [2]~~

~~catanh(GLIBC_2.2) [2]~~	~~ctanhf(GLIBC_2.2) [2]~~	~~hypot(GLIBC_2.2) [2]~~	~~nextafter(GLIBC_2.2) [2]~~	~~y0f(GLIBC_2.2) [1]~~
~~catanhf(GLIBC_2.2) [2]~~	~~ctanhl(GLIBC_2.2) [2]~~	~~hypotf(GLIBC_2.2) [2]~~	~~nextafterf(GLIBC_2.2) [2]~~	~~y0l(GLIBC_2.2) [1]~~
~~catanhl(GLIBC_2.2) [2]~~	~~ctanl(GLIBC_2.2) [2]~~	~~hypotl(GLIBC_2.2) [2]~~	~~nextafterl(GLIBC_2.2) [2]~~	~~y1(GLIBC_2.2) [2]~~
~~catanl(GLIBC_2.2) [2]~~	~~dremf(GLIBC_2.2) [1]~~	~~ilogb(GLIBC_2.2) [2]~~	~~nexttoward(GLIBC_2.2) [2]~~	~~y1f(GLIBC_2.2) [1]~~
~~cbrt(GLIBC_2.2) [2]~~	~~dreml(GLIBC_2.2) [1]~~	~~ilogbf(GLIBC_2.2) [2]~~	~~nexttowardf(GLIBC_2.2) [2]~~	~~y1l(GLIBC_2.2) [1]~~
~~cbrtf(GLIBC_2.2) [2]~~	~~erf(GLIBC_2.2) [2]~~	~~ilogbl(GLIBC_2.2) [2]~~	~~nexttowardl(GLIBC_2.2) [2]~~	~~yn(GLIBC_2.2) [2]~~
~~cbrtl(GLIBC_2.2) [2]~~	~~erfc(GLIBC_2.2) [2]~~	~~j0(GLIBC_2.2) [2]~~	~~pow(GLIBC_2.2) [2]~~	~~ynf(GLIBC_2.2) [1]~~
~~ccos(GLIBC_2.2) [2]~~	~~erfcf(GLIBC_2.2) [2]~~	~~j0f(GLIBC_2.2) [1]~~	~~pow10(GLIBC_2.2) [1]~~	~~ynl(GLIBC_2.2) [1]~~
~~ccosf(GLIBC_2.2) [2]~~	~~erfcl(GLIBC_2.2) [2]~~	~~j0l(GLIBC_2.2) [1]~~	~~pow10f(GLIBC_2.2) [1]~~	
~~ccosh(GLIBC_2.2) [2]~~	~~erff(GLIBC_2.2) [2]~~	~~j1(GLIBC_2.2) [2]~~	~~pow10l(GLIBC_2.2) [1]~~	
~~ccoshf(GLIBC_2.2) [2]~~	~~erfl(GLIBC_2.2) [2]~~	~~j1f(GLIBC_2.2) [1]~~	~~powf(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].**~~

__finite(GLIBC_2.2) [ISOC99]	__finitef(GLIBC_2.2) [ISOC99]	__finitel(GLIBC_2.2) [ISOC99]	__fpclassify(GLIBC_2.2) [LSB]
__fpclassifyf(GLIBC_2.2) [LSB]	__fpclassifyl(GLIBC_2.2) [ISOC99]	__signbit(GLIBC_2.2) [ISOC99]	__signbitf(GLIBC_2.2) [ISOC99]
__signbitl(GLIBC_2.2) [ISOC99]	acos(GLIBC_2.2) [SUSv3]	acosf(GLIBC_2.2) [SUSv3]	acosh(GLIBC_2.2) [SUSv3]
acoshf(GLIBC_2.2) [SUSv3]	acoshl(GLIBC_2.2) [SUSv3]	acosl(GLIBC_2.2) [SUSv3]	asin(GLIBC_2.2) [SUSv3]
asinf(GLIBC_2.2) [SUSv3]	asinh(GLIBC_2.2) [SUSv3]	asinhf(GLIBC_2.2) [SUSv3]	asinhl(GLIBC_2.2) [SUSv3]
asinl(GLIBC_2.2) [SUSv3]	atan(GLIBC_2.2) [SUSv3]	atan2(GLIBC_2.2) [SUSv3]	atan2f(GLIBC_2.2) [SUSv3]
atan2l(GLIBC_2.2) [SUSv3]	atanf(GLIBC_2.2) [SUSv3]	atanh(GLIBC_2.2) [SUSv3]	atanhf(GLIBC_2.2) [SUSv3]
atanhl(GLIBC_2.2	atanl(GLIBC_2.2)	cabs(GLIBC_2.2)	cabsf(GLIBC_2.2)

) [SUSv3]	[SUSv3]	[SUSv3]	[SUSv3]
cabsl(GLIBC_2.2) [SUSv3]	cacos(GLIBC_2.2) [SUSv3]	cacosf(GLIBC_2.2) [SUSv3]	cacosh(GLIBC_2.2) [SUSv3]
cacoshf(GLIBC_2. 2) [SUSv3]	cacoshl(GLIBC_2. 2) [SUSv3]	cacosl(GLIBC_2.2) [SUSv3]	carg(GLIBC_2.2) [SUSv3]
cargf(GLIBC_2.2) [SUSv3]	cargl(GLIBC_2.2) [SUSv3]	casin(GLIBC_2.2) [SUSv3]	casinf(GLIBC_2.2) [SUSv3]
casinh(GLIBC_2.2) [SUSv3]	casinhf(GLIBC_2. 2) [SUSv3]	casinhl(GLIBC_2. 2) [SUSv3]	casinl(GLIBC_2.2) [SUSv3]
catan(GLIBC_2.2) [SUSv3]	catanf(GLIBC_2.2) [SUSv3]	catanh(GLIBC_2.2) [SUSv3]	catanhf(GLIBC_2. 2) [SUSv3]
catanhl(GLIBC_2. 2) [SUSv3]	catanl(GLIBC_2.2) [SUSv3]	cbrt(GLIBC_2.2) [SUSv3]	cbrtf(GLIBC_2.2) [SUSv3]
cbrtl(GLIBC_2.2) [SUSv3]	ccos(GLIBC_2.2) [SUSv3]	ccosf(GLIBC_2.2) [SUSv3]	ccosh(GLIBC_2.2) [SUSv3]
ccoshf(GLIBC_2.2) [SUSv3]	ccoshl(GLIBC_2.2) [SUSv3]	ccosl(GLIBC_2.2) [SUSv3]	ceil(GLIBC_2.2) [SUSv3]
ceilf(GLIBC_2.2) [SUSv3]	ceill(GLIBC_2.2) [SUSv3]	cexp(GLIBC_2.2) [SUSv3]	cexpf(GLIBC_2.2) [SUSv3]
cexpl(GLIBC_2.2) [SUSv3]	cimag(GLIBC_2.2) [SUSv3]	cimagf(GLIBC_2.2) [SUSv3]	cimagl(GLIBC_2.2) [SUSv3]
clog(GLIBC_2.2) [SUSv3]	clog10(GLIBC_2.2) [ISOC99]	clog10f(GLIBC_2. 2) [ISOC99]	clog10l(GLIBC_2. 2) [ISOC99]
clogf(GLIBC_2.2) [SUSv3]	clogl(GLIBC_2.2) [SUSv3]	conj(GLIBC_2.2) [SUSv3]	conjf(GLIBC_2.2) [SUSv3]
conjl(GLIBC_2.2) [SUSv3]	copysign(GLIBC_ 2.2) [SUSv3]	copysignf(GLIBC_ 2.2) [SUSv3]	copysignl(GLIBC_ 2.2) [SUSv3]
cos(GLIBC_2.2) [SUSv3]	cosf(GLIBC_2.2) [SUSv3]	cosh(GLIBC_2.2) [SUSv3]	coshf(GLIBC_2.2) [SUSv3]
coshl(GLIBC_2.2) [SUSv3]	cosl(GLIBC_2.2) [SUSv3]	cpow(GLIBC_2.2) [SUSv3]	cpowf(GLIBC_2.2) [SUSv3]
cpowl(GLIBC_2.2) [SUSv3]	cproj(GLIBC_2.2) [SUSv3]	cprojf(GLIBC_2.2) [SUSv3]	cprojl(GLIBC_2.2) [SUSv3]
creal(GLIBC_2.2) [SUSv3]	crealf(GLIBC_2.2) [SUSv3]	creall(GLIBC_2.2) [SUSv3]	csin(GLIBC_2.2) [SUSv3]
csinf(GLIBC_2.2) [SUSv3]	csinh(GLIBC_2.2) [SUSv3]	csinhf(GLIBC_2.2) [SUSv3]	csinhl(GLIBC_2.2) [SUSv3]
csinl(GLIBC_2.2) [SUSv3]	csqrt(GLIBC_2.2) [SUSv3]	csqrtf(GLIBC_2.2) [SUSv3]	csqrtl(GLIBC_2.2) [SUSv3]
ctan(GLIBC_2.2) [SUSv3]	ctanf(GLIBC_2.2) [SUSv3]	ctanh(GLIBC_2.2) [SUSv3]	ctanhf(GLIBC_2.2) [SUSv3]

ctanhl(GLIBC_2.2) [SUSv3]	ctanl(GLIBC_2.2) [SUSv3]	dremf(GLIBC_2.2) [ISOC99]	dreml(GLIBC_2.2) [ISOC99]
erf(GLIBC_2.2) [SUSv3]	erfc(GLIBC_2.2) [SUSv3]	erfcf(GLIBC_2.2) [SUSv3]	erfcl(GLIBC_2.2) [SUSv3]
erff(GLIBC_2.2) [SUSv3]	erfl(GLIBC_2.2) [SUSv3]	exp(GLIBC_2.2) [SUSv3]	exp2(GLIBC_2.2) [SUSv3]
exp2f(GLIBC_2.2) [SUSv3]	exp2l(GLIBC_2.2) [SUSv3]	expf(GLIBC_2.2) [SUSv3]	expl(GLIBC_2.2) [SUSv3]
expm1(GLIBC_2.2) [SUSv3]	expm1f(GLIBC_2.2) [SUSv3]	expm1l(GLIBC_2.2) [SUSv3]	fabs(GLIBC_2.2) [SUSv3]
fabsf(GLIBC_2.2) [SUSv3]	fabsl(GLIBC_2.2) [SUSv3]	fdim(GLIBC_2.2) [SUSv3]	fdimf(GLIBC_2.2) [SUSv3]
fdiml(GLIBC_2.2) [SUSv3]	feclearexcept(GLIBC_2.2) [SUSv3]	fegetenv(GLIBC_2.2) [SUSv3]	fegetexceptflag(GLIBC_2.2) [SUSv3]
fegetround(GLIBC_2.2) [SUSv3]	feholdexcept(GLIBC_2.2) [SUSv3]	feraiseexcept(GLIBC_2.2) [SUSv3]	fesetenv(GLIBC_2.2) [SUSv3]
fesetexceptflag(GLIBC_2.2) [SUSv3]	fesetround(GLIBC_2.2) [SUSv3]	fetestexcept(GLIBC_2.2) [SUSv3]	feupdateenv(GLIBC_2.2) [SUSv3]
finite(GLIBC_2.2) [SUSv2]	finitef(GLIBC_2.2) [ISOC99]	finitel(GLIBC_2.2) [ISOC99]	floor(GLIBC_2.2) [SUSv3]
floorf(GLIBC_2.2) [SUSv3]	floorl(GLIBC_2.2) [SUSv3]	fma(GLIBC_2.2) [SUSv3]	fmaf(GLIBC_2.2) [SUSv3]
fmal(GLIBC_2.2) [SUSv3]	fmax(GLIBC_2.2) [SUSv3]	fmaxf(GLIBC_2.2) [SUSv3]	fmaxl(GLIBC_2.2) [SUSv3]
fmin(GLIBC_2.2) [SUSv3]	fminf(GLIBC_2.2) [SUSv3]	fminl(GLIBC_2.2) [SUSv3]	fmod(GLIBC_2.2) [SUSv3]
fmodf(GLIBC_2.2) [SUSv3]	fmodl(GLIBC_2.2) [SUSv3]	frexp(GLIBC_2.2) [SUSv3]	frexpf(GLIBC_2.2) [SUSv3]
frexpl(GLIBC_2.2) [SUSv3]	gamma(GLIBC_2.2) [SUSv2]	gammaf(GLIBC_2.2) [ISOC99]	gammal(GLIBC_2.2) [ISOC99]
hypot(GLIBC_2.2) [SUSv3]	hypotf(GLIBC_2.2) [SUSv3]	hypotl(GLIBC_2.2) [SUSv3]	ilogb(GLIBC_2.2) [SUSv3]
ilogbf(GLIBC_2.2) [SUSv3]	ilogbl(GLIBC_2.2) [SUSv3]	j0(GLIBC_2.2) [SUSv3]	j0f(GLIBC_2.2) [ISOC99]
j0l(GLIBC_2.2) [ISOC99]	j1(GLIBC_2.2) [SUSv3]	j1f(GLIBC_2.2) [ISOC99]	j1l(GLIBC_2.2) [ISOC99]
jn(GLIBC_2.2) [SUSv3]	jnf(GLIBC_2.2) [ISOC99]	jnl(GLIBC_2.2) [ISOC99]	ldexp(GLIBC_2.2) [SUSv3]
ldexpf(GLIBC_2.2) [SUSv3]	ldexpl(GLIBC_2.2) [SUSv3]	lgamma(GLIBC_2.2) [SUSv3]	lgamma_r(GLIBC_2.2) [ISOC99]
lgammaf(GLIBC_	lgammaf_r(GLIB	lgammal(GLIBC_	lgammal_r(GLIBC

2.2) [SUSv3]	C_2.2) [ISOC99]	2.2) [SUSv3]	_2.2) [ISOC99]
llrint(GLIBC_2.2) [SUSv3]	llrintf(GLIBC_2.2) [SUSv3]	llrintl(GLIBC_2.2) [SUSv3]	llround(GLIBC_2.2) [SUSv3]
llroundf(GLIBC_2.2) [SUSv3]	llroundl(GLIBC_2.2) [SUSv3]	log(GLIBC_2.2) [SUSv3]	log10(GLIBC_2.2) [SUSv3]
log10f(GLIBC_2.2) [SUSv3]	log10l(GLIBC_2.2) [SUSv3]	log1p(GLIBC_2.2) [SUSv3]	log1pf(GLIBC_2.2) [SUSv3]
log1pl(GLIBC_2.2) [SUSv3]	log2(GLIBC_2.2) [SUSv3]	log2f(GLIBC_2.2) [SUSv3]	log2l(GLIBC_2.2) [SUSv3]
logb(GLIBC_2.2) [SUSv3]	logbf(GLIBC_2.2) [SUSv3]	logbl(GLIBC_2.2) [SUSv3]	logf(GLIBC_2.2) [SUSv3]
logl(GLIBC_2.2) [SUSv3]	lrint(GLIBC_2.2) [SUSv3]	lrintf(GLIBC_2.2) [SUSv3]	lrintl(GLIBC_2.2) [SUSv3]
lround(GLIBC_2.2) [SUSv3]	lroundf(GLIBC_2.2) [SUSv3]	lroundl(GLIBC_2.2) [SUSv3]	matherr(GLIBC_2.2) [ISOC99]
modf(GLIBC_2.2) [SUSv3]	modff(GLIBC_2.2) [SUSv3]	modfl(GLIBC_2.2) [SUSv3]	nan(GLIBC_2.2) [SUSv3]
nanf(GLIBC_2.2) [SUSv3]	nanl(GLIBC_2.2) [SUSv3]	nearbyint(GLIBC_2.2) [SUSv3]	nearbyintf(GLIBC_2.2) [SUSv3]
nearbyintl(GLIBC_2.2) [SUSv3]	nextafter(GLIBC_2.2) [SUSv3]	nextafterf(GLIBC_2.2) [SUSv3]	nextafterl(GLIBC_2.2) [SUSv3]
nexttoward(GLIBC_2.2) [SUSv3]	nexttowardf(GLIBC_2.2) [SUSv3]	nexttowardl(GLIBC_2.2) [SUSv3]	pow(GLIBC_2.2) [SUSv3]
pow10(GLIBC_2.2) [ISOC99]	pow10f(GLIBC_2.2) [ISOC99]	pow10l(GLIBC_2.2) [ISOC99]	powf(GLIBC_2.2) [SUSv3]
powl(GLIBC_2.2) [SUSv3]	remainder(GLIBC_2.2) [SUSv3]	remainderf(GLIBC_2.2) [SUSv3]	remainderl(GLIBC_2.2) [SUSv3]
remquo(GLIBC_2.2) [SUSv3]	remquof(GLIBC_2.2) [SUSv3]	remquol(GLIBC_2.2) [SUSv3]	rint(GLIBC_2.2) [SUSv3]
rintf(GLIBC_2.2) [SUSv3]	rintl(GLIBC_2.2) [SUSv3]	round(GLIBC_2.2) [SUSv3]	roundf(GLIBC_2.2) [SUSv3]
roundl(GLIBC_2.2) [SUSv3]	scalb(GLIBC_2.2) [SUSv3]	scalbf(GLIBC_2.2) [ISOC99]	scalbl(GLIBC_2.2) [ISOC99]
scalbln(GLIBC_2.2) [SUSv3]	scalblnf(GLIBC_2.2) [SUSv3]	scalblnl(GLIBC_2.2) [SUSv3]	scalbn(GLIBC_2.2) [SUSv3]
scalbnf(GLIBC_2.2) [SUSv3]	scalbnl(GLIBC_2.2) [SUSv3]	significand(GLIBC_2.2) [ISOC99]	significandf(GLIBC_2.2) [ISOC99]
significandl(GLIBC_2.2) [ISOC99]	sin(GLIBC_2.2) [SUSv3]	sincos(GLIBC_2.2) [ISOC99]	sincosf(GLIBC_2.2) [ISOC99]
sincosl(GLIBC_2.2) [ISOC99]	sinf(GLIBC_2.2) [SUSv3]	sinh(GLIBC_2.2) [SUSv3]	sinhf(GLIBC_2.2) [SUSv3]

sinhl(GLIBC_2.2) [SUSv3]	sinl(GLIBC_2.2) [SUSv3]	sqrt(GLIBC_2.2) [SUSv3]	sqrtf(GLIBC_2.2) [SUSv3]
sqrtl(GLIBC_2.2) [SUSv3]	tan(GLIBC_2.2) [SUSv3]	tanf(GLIBC_2.2) [SUSv3]	tanh(GLIBC_2.2) [SUSv3]
tanhf(GLIBC_2.2) [SUSv3]	tanhl(GLIBC_2.2) [SUSv3]	tanl(GLIBC_2.2) [SUSv3]	tgamma(GLIBC_2.2) [SUSv3]
tgammaf(GLIBC_2.2) [SUSv3]	tgammal(GLIBC_2.2) [SUSv3]	trunc(GLIBC_2.2) [SUSv3]	truncf(GLIBC_2.2) [SUSv3]
truncl(GLIBC_2.2) [SUSv3]	y0(GLIBC_2.2) [SUSv3]	y0f(GLIBC_2.2) [ISOC99]	y0l(GLIBC_2.2) [ISOC99]
y1(GLIBC_2.2) [SUSv3]	y1f(GLIBC_2.2) [ISOC99]	y1l(GLIBC_2.2) [ISOC99]	yn(GLIBC_2.2) [SUSv3]
ynf(GLIBC_2.2) [ISOC99]	ynl(GLIBC_2.2) [ISOC99]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Math specified in ~~ISO C (1999)~~Table 11-26

~~[2]. ISO POSIX (2003)~~

~~[3]. this specification~~

~~[4]. SUSv2~~

~~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_2.2) [1]~~				

~~*Referenced Specification(s)*~~

~~[1]. ISO POSIX (2003)~~

signgam(GLIBC_2.2) [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.

~~These definitions are intended to supplement those provided in~~ Where an interface is defined as requiring a particular system header file all of the ~~referenced underlying~~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 ~~ISO/IEC 9899~~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

```

extern double cabs(double complex);
extern float cabsf(float complex);
extern long double cabsl(long double complex);
extern double complex cacos(double complex);
extern float complex cacosf(float complex);
extern double complex cacosh(double complex);
extern float complex cacoshf(float complex);
extern long double complex cacoshl(long double complex);
extern long double complex cacosl(long double complex);
extern double carg(double complex);
extern float cargf(float complex);
extern long double cargl(long double complex);
extern double complex casin(double complex);
extern float complex casinf(float complex);
extern double complex casinh(double complex);
extern float complex casinhf(float complex);
extern long double complex casinhl(long double complex);
extern long double complex casinl(long double complex);
extern double complex catan(double complex);
extern float complex catanf(float complex);
extern double complex catanh(double complex);
extern float complex catanhf(float complex);
extern long double complex catanhl(long double complex);
extern long double complex catanl(long double complex);
extern double complex ccos(double complex);
extern float complex ccosf(float complex);
extern double complex ccosh(double complex);
extern float complex ccoshf(float complex);
extern long double complex ccoshl(long double complex);
extern long double complex ccosl(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexpl(long double complex);
extern double cimag(double complex);
extern float cimagf(float complex);
extern long double cimagl(long double complex);
extern double complex clog(double complex);
extern float complex clog10f(float complex);
extern long double complex clog10l(long double complex);
extern float complex clogf(float complex);
extern long double complex clogl(long double complex);
extern double complex conj(double complex);
extern float complex conjf(float complex);
extern long double complex conjl(long double complex);
extern double complex cpow(double complex, double complex);
extern float complex cpowf(float complex, float complex);
extern long double complex cpowl(long double complex, long double
complex);
extern double complex cproj(double complex);
extern float complex cprojf(float complex);
extern long double complex cprojl(long double complex);
extern double creal(double complex);
extern float crealf(float complex);
extern long double creall(long double complex);
extern double complex csin(double complex);
extern float complex csinf(float complex);
extern double complex csinh(double complex);
extern float complex csinhf(float complex);
extern long double complex csinhl(long double complex);
extern long double complex csinl(long double complex);
extern double complex csqrt(double complex);
extern float complex csqrtf(float complex);
extern long double complex csqrtl(long double complex);
extern double complex ctan(double complex);
extern float complex ctanf(float complex);
extern double complex ctanh(double complex);
extern float complex ctanhf(float complex);
extern long double complex ctanhl(long double complex);
extern long double complex ctanl(long double complex);
```

11.5.2 fenv.h

```

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

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

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

typedef unsigned long int fexcept_t;

typedef unsigned long int fenv_t;

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

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

11.5.~~2~~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       -2147483648
#define FP_ILOGBNAN     2147483647

extern int __finite(double);
extern int __finitef(float);
extern int __finitel(long double);
extern int __isinf(double);
extern int __isinff(float);
extern int __isinfl(long double);
extern int __isnan(double);
extern int __isnanf(float);
extern int __isnanl(long double);
extern int __signbit(double);
extern int __signbitf(float);
extern int __fpclassify(double);
extern int __fpclassifyf(float);
extern int __fpclassifyl(long double);
extern int signgam(void);
extern double copysign(double, double);
extern int finite(double);
extern double frexp(double, int *);
extern double ldexp(double, int);
extern double modf(double, double *);
extern double acos(double);
extern double acosh(double);
extern double asinh(double);
extern double atanh(double);
extern double asin(double);
extern double atan(double);
extern double atan2(double, double);
extern double cbrt(double);
extern double ceil(double);
extern double cos(double);
extern double cosh(double);
extern double erf(double);
extern double erfc(double);
extern double exp(double);
extern double expm1(double);
extern double fabs(double);
extern double floor(double);
extern double fmod(double, double);
extern double gamma(double);
extern double hypot(double, double);
extern int ilogb(double);
extern double j0(double);
extern double j1(double);
extern double jn(int, double);
extern double lgamma(double);
extern double log(double);
extern double log10(double);
extern double log1p(double);
extern double logb(double);
extern double nextafter(double, double);
extern double pow(double, double);
extern double remainder(double, double);
extern double rint(double);
extern double scalb(double, double);
extern double sin(double);
extern double sinh(double);
extern double sqrt(double);
extern double tan(double);
extern double tanh(double);
extern double y0(double);
```

```
          extern double y1(double);
          extern double yn(int, double);
          extern float copysignf(float, float);
          extern long double copysignl(long double, long double);
          extern int finitef(float);
          extern int finitel(long double);
          extern float frexpf(float, int *);
          extern long double frexpl(long double, int *);
          extern float ldexpf(float, int);
          extern long double ldexpl(long double, int);
          extern float modff(float, float *);
          extern long double modfl(long double, long double *);
          extern double scalbln(double, long int);
          extern float scalblnf(float, long int);
          extern long double scalblnl(long double, long int);
          extern double scalbn(double, int);
          extern float scalbnf(float, int);
          extern long double scalbnl(long double, int);
          extern float acosf(float);
          extern float acoshf(float);
          extern long double acoshl(long double);
          extern long double acosl(long double);
          extern float asinf(float);
          extern float asinhf(float);
          extern long double asinhl(long double);
          extern long double asinl(long double);
          extern float atan2f(float, float);
          extern long double atan2l(long double, long double);
          extern float atanf(float);
          extern float atanhf(float);
          extern long double atanhl(long double);
          extern long double atanl(long double);
          extern float cbrtf(float);
          extern long double cbrtl(long double);
          extern float ceilf(float);
          extern long double ceill(long double);
          extern float cosf(float);
          extern float coshf(float);
          extern long double coshl(long double);
          extern long double cosl(long double);
          extern float dremf(float, float);
          extern long double dreml(long double, long double);
          extern float erfcf(float);
          extern long double erfcl(long double);
          extern float erff(float);
          extern long double erfl(long double);
          extern double exp2(double);
          extern float exp2f(float);
          extern long double exp2l(long double);
          extern float expf(float);
          extern long double expl(long double);
          extern float expm1f(float);
          extern long double expm1l(long double);
          extern float fabsf(float);
          extern long double fabsl(long double);
          extern double fdim(double, double);
          extern float fdimf(float, float);
          extern long double fdiml(long double, long double);
          extern float floorf(float);
          extern long double floorl(long double);
          extern double fma(double, double, double);
          extern float fmaf(float, float, float);
          extern long double fmal(long double, long double, long double);
          extern double fmax(double, double);
          extern float fmaxf(float, float);
          extern long double fmaxl(long double, long double);
          extern double fmin(double, double);
          extern float fminf(float, float);
          extern long double fminl(long double, long double);
          extern float fmodf(float, float);
          extern long double fmodl(long double, long double);
          extern float gammaf(float);
          extern long double gammal(long double);
          extern float hypotf(float, float);
          extern long double hypotl(long double, long double);
          extern int ilogbf(float);
          extern int ilogbl(long double);
          extern float j0f(float);
          extern long double j0l(long double);
          extern float j1f(float);
          extern long double j1l(long double);
          extern float jnf(int, float);
          extern long double jnl(int, long double);
          extern double lgamma_r(double, int *);
          extern float lgammaf(float);
          extern float lgammaf_r(float, int *);
          extern long double lgammal(long double);
          extern long double lgammal_r(long double, int *);
          extern long long int llrint(double);
          extern long long int llrintf(float);
          extern long long int llrintl(long double);
          extern long long int llround(double);
          extern long long int llroundf(float);
          extern long long int llroundl(long double);
          extern float log10f(float);
          extern long double log10l(long double);
          extern float log1pf(float);
          extern long double log1pl(long double);
          extern double log2(double);
          extern float log2f(float);
          extern long double log2l(long double);
          extern float logbf(float);
          extern long double logbl(long double);
          extern float logf(float);
          extern long double logl(long double);
          extern long int lrint(double);
          extern long int lrintf(float);
          extern long int lrintl(long double);
          extern long int lround(double);
          extern long int lroundf(float);
          extern long int lroundl(long double);
          extern int matherr(struct exception *);
          extern double nan(const char *);
          extern float nanf(const char *);
          extern long double nanl(const char *);
          extern double nearbyint(double);
          extern float nearbyintf(float);
          extern long double nearbyintl(long double);
          extern float nextafterf(float, float);
          extern long double nextafterl(long double, long double);
          extern double nexttoward(double, long double);
          extern float nexttowardf(float, long double);
          extern long double nexttowardl(long double, long double);
          extern double pow10(double);
          extern float pow10f(float);
          extern long double pow10l(long double);
          extern float powf(float, float);
          extern long double powl(long double, long double);
```

```
extern float remainderf(float, float);
extern long double remainderl(long double, long double);
extern double remquo(double, double, int *);
extern float remquof(float, float, int *);
extern long double remquol(long double, long double, int *);
extern float rintf(float);
extern long double rintl(long double);
extern double round(double);
extern float roundf(float);
extern long double roundl(long double);
extern float scalbf(float, float);
extern long double scalbl(long double, long double);
extern double significand(double);
extern float significandf(float);
extern long double significandl(long double);
extern void sincos(double, double *, double *);
extern void sincosf(float, float *, float *);
extern void sincosl(long double, long double *, long double *);
extern float sinf(float);
extern float sinhf(float);
extern long double sinhl(long double);
extern long double sinl(long double);
extern float sqrtf(float);
extern long double sqrtl(long double);
extern float tanf(float);
extern float tanhf(float);
extern long double tanhl(long double);
extern long double tanl(long double);
extern double tgamma(double);
extern float tgammaf(float);
extern long double tgammal(long double);
extern double trunc(double);
extern float truncf(float);
extern long double truncl(long double);
extern float y0f(float);
extern long double y0l(long double);
extern float y1f(float);
extern long double y1l(long double);
extern float ynf(int, float);
extern long double ynl(int, long double);
extern int __fpclassifyl(long double);
extern int __fpclassifyl(long double);
extern int __signbitl(long double);
extern int __signbitl(long double);
extern int __signbitl(long double);
extern long double exp2l(long double);
extern long double exp2l(long double);
```

11.6 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] ~~this specification~~This Specification
[SUSv3] ISO POSIX (2003)

11.6.1 Realtime Threads

11.6.1.1 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_2.2) [1]~~	~~pthread_attr_getscope(GLIBC_2.2) [1]~~	~~pthread_attr_setschedpolicy(GLIBC_2.2) [1]~~	~~pthread_getschedparam(GLIBC_2.2) [1]~~	
~~pthread_attr_getschedpolicy(GLIBC_2.2) [1]~~	~~pthread_attr_setinheritsched(GLIBC_2.2) [1]~~	~~pthread_attr_setscope(GLIBC_2.2) [1]~~	~~pthread_setschedparam(GLIBC_2.2) [1]~~	

~~*Referenced Specification(s)*~~

~~**[1].** ISO POSIX (2003)~~

pthread_attr_getinheritsched(GLIBC_2.2) [SUSv3]	pthread_attr_getschedpolicy(GLIBC_2.2) [SUSv3]	pthread_attr_getscope(GLIBC_2.2) [SUSv3]	pthread_attr_setinheritsched(GLIBC_2.2) [SUSv3]
pthread_attr_setschedpolicy(GLIBC_2.2) [SUSv3]	pthread_attr_setscope(GLIBC_2.2) [SUSv3]	pthread_getschedparam(GLIBC_2.2) [SUSv3]	pthread_setschedparam(GLIBC_2.2) [SUSv3]

11.6.2 Advanced Realtime Threads

11.6.2.1 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.

11.6.3 Posix Threads

11.6.3.1 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_2.2) [1]~~	~~pthread_cond_broadcast(GLIBC_2.3.2) [2]~~	~~pthread_join(GLIBC_2.2) [2]~~	~~pthread_rwlock_destroy(GLIBC_2.2) [2]~~	~~pthread_setconcurrency(GLIBC_2.2) [2]~~
~~_pthread_clea~~	~~pthread_cond~~	~~pthread_key_~~	~~pthread_rwlo~~	~~pthread_setsp~~

~~nup_push(GLIBC_2.2) [1]~~	~~_destroy(GLIBC_2.3.2) [2]~~	~~create(GLIBC_2.2) [2]~~	~~ck_init(GLIBC_2.2) [2]~~	~~ecific(GLIBC_2.2) [2]~~
~~pthread_attr_destroy(GLIBC_2.2) [2]~~	~~pthread_cond_init(GLIBC_2.3.2) [2]~~	~~pthread_key_delete(GLIBC_2.2) [2]~~	~~pthread_rwlock_rdlock(GLIBC_2.2) [2]~~	~~pthread_sigmask(GLIBC_2.2) [2]~~
~~pthread_attr_getdetachstate(GLIBC_2.2) [2]~~	~~pthread_cond_signal(GLIBC_2.3.2) [2]~~	~~pthread_kill(GLIBC_2.2) [2]~~	~~pthread_rwlock_timedrdlock(GLIBC_2.2) [2]~~	~~pthread_testcancel(GLIBC_2.2) [2]~~
~~pthread_attr_getguardsize(GLIBC_2.2) [2]~~	~~pthread_cond_timedwait(GLIBC_2.3.2) [2]~~	~~pthread_mutex_destroy(GLIBC_2.2) [2]~~	~~pthread_rwlock_timedwrlock(GLIBC_2.2) [2]~~	~~sem_close(GLIBC_2.2) [2]~~
~~pthread_attr_getschedparam(GLIBC_2.2) [2]~~	~~pthread_cond_wait(GLIBC_2.3.2) [2]~~	~~pthread_mutex_init(GLIBC_2.2) [2]~~	~~pthread_rwlock_tryrdlock(GLIBC_2.2) [2]~~	~~sem_destroy(GLIBC_2.2) [2]~~
~~pthread_attr_getstack(GLIBC_2.2) [2]~~	~~pthread_condattr_destroy(GLIBC_2.2) [2]~~	~~pthread_mutex_lock(GLIBC_2.2) [2]~~	~~pthread_rwlock_trywrlock(GLIBC_2.2) [2]~~	~~sem_getvalue(GLIBC_2.2) [2]~~
~~pthread_attr_getstackaddr(GLIBC_2.2) [2]~~	~~pthread_condattr_getpshared(GLIBC_2.2) [2]~~	~~pthread_mutex_trylock(GLIBC_2.2) [2]~~	~~pthread_rwlock_unlock(GLIBC_2.2) [2]~~	~~sem_init(GLIBC_2.2) [2]~~
~~pthread_attr_getstacksize(GLIBC_2.2) [2]~~	~~pthread_condattr_init(GLIBC_2.2) [2]~~	~~pthread_mutex_unlock(GLIBC_2.2) [2]~~	~~pthread_rwlock_wrlock(GLIBC_2.2) [2]~~	~~sem_open(GLIBC_2.2) [2]~~
~~pthread_attr_init(GLIBC_2.2) [2]~~	~~pthread_condattr_setpshared(GLIBC_2.2) [2]~~	~~pthread_mutexattr_destroy(GLIBC_2.2) [2]~~	~~pthread_rwlockattr_destroy(GLIBC_2.2) [2]~~	~~sem_post(GLIBC_2.2) [2]~~
~~pthread_attr_setdetachstate(GLIBC_2.2) [2]~~	~~pthread_create(GLIBC_2.2) [2]~~	~~pthread_mutexattr_getpshared(GLIBC_2.2) [2]~~	~~pthread_rwlockattr_getpshared(GLIBC_2.2) [2]~~	~~sem_timedwait(GLIBC_2.2) [2]~~
~~pthread_attr_setguardsize(GLIBC_2.2) [2]~~	~~pthread_detach(GLIBC_2.2) [2]~~	~~pthread_mutexattr_gettype(GLIBC_2.2) [2]~~	~~pthread_rwlockattr_init(GLIBC_2.2) [2]~~	~~sem_trywait(GLIBC_2.2) [2]~~
~~pthread_attr_setschedparam(GLIBC_2.2) [2]~~	~~pthread_equal(GLIBC_2.2) [2]~~	~~pthread_mutexattr_init(GLIBC_2.2) [2]~~	~~pthread_rwlockattr_setpshared(GLIBC_2.2) [2]~~	~~sem_unlink(GLIBC_2.2) [2]~~
~~pthread_attr_~~	~~pthread_exit(~~	~~pthread_mute~~	~~pthread_self(~~	~~sem_wait(GLI~~

~~setstackaddr(GLIBC_2.2) [2]~~	~~GLIBC_2.2) [2]~~	~~xattr_setpshared(GLIBC_2.2) [2]~~	~~GLIBC_2.2) [2]~~	~~BC_2.2) [2]~~
~~pthread_attr_setstacksize(GLIBC_2.3.3) [2]~~	~~pthread_getconcurrency(GLIBC_2.2) [2]~~	~~pthread_mutexattr_settype(GLIBC_2.2) [2]~~	~~pthread_setcancelstate(GLIBC_2.2) [2]~~	
~~pthread_cancel(GLIBC_2.2) [2]~~	~~pthread_getspecific(GLIBC_2.2) [2]~~	~~pthread_once(GLIBC_2.2) [2]~~	~~pthread_setcanceltype(GLIBC_2.2) [2]~~	

~~*Referenced Specification(s)*~~

~~**[1].** this specification~~

~~**[2].** ISO POSIX (2003)~~

_pthread_cleanup_pop(GLIBC_2.2) [LSB]	_pthread_cleanup_push(GLIBC_2.2) [LSB]	pthread_attr_destroy(GLIBC_2.2) [SUSv3]	pthread_attr_getdetachstate(GLIBC_2.2) [SUSv3]
pthread_attr_getguardsize(GLIBC_2.2) [SUSv3]	pthread_attr_getschedparam(GLIBC_2.2) [SUSv3]	pthread_attr_getstack(GLIBC_2.2) [SUSv3]	pthread_attr_getstackaddr(GLIBC_2.2) [SUSv3]
pthread_attr_getstacksize(GLIBC_2.2) [SUSv3]	pthread_attr_init(GLIBC_2.2) [SUSv3]	pthread_attr_setdetachstate(GLIBC_2.2) [SUSv3]	pthread_attr_setguardsize(GLIBC_2.2) [SUSv3]
pthread_attr_setschedparam(GLIBC_2.2) [SUSv3]	pthread_attr_setstackaddr(GLIBC_2.2) [SUSv3]	pthread_attr_setstacksize(GLIBC_2.3.3) [SUSv3]	pthread_cancel(GLIBC_2.2) [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_2.2) [SUSv3]	pthread_condattr_getpshared(GLIBC_2.2) [SUSv3]
pthread_condattr_init(GLIBC_2.2) [SUSv3]	pthread_condattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_create(GLIBC_2.2) [SUSv3]	pthread_detach(GLIBC_2.2) [SUSv3]
pthread_equal(GLIBC_2.2) [SUSv3]	pthread_exit(GLIBC_2.2) [SUSv3]	pthread_getconcurrency(GLIBC_2.2) [SUSv3]	pthread_getspecific(GLIBC_2.2) [SUSv3]
pthread_join(GLIBC_2.2) [SUSv3]	pthread_key_create(GLIBC_2.2) [SUSv3]	pthread_key_delete(GLIBC_2.2) [SUSv3]	pthread_kill(GLIBC_2.2) [SUSv3]
pthread_mutex_destroy(GLIBC_2.2) [SUSv3]	pthread_mutex_init(GLIBC_2.2) [SUSv3]	pthread_mutex_lock(GLIBC_2.2) [SUSv3]	pthread_mutex_trylock(GLIBC_2.2) [SUSv3]

pthread_mutex_unlock(GLIBC_2.2) [SUSv3]	pthread_mutexattr_destroy(GLIBC_2.2) [SUSv3]	pthread_mutexattr_getpshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_gettype(GLIBC_2.2) [SUSv3]
pthread_mutexattr_init(GLIBC_2.2) [SUSv3]	pthread_mutexattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_settype(GLIBC_2.2) [SUSv3]	pthread_once(GLIBC_2.2) [SUSv3]
pthread_rwlock_destroy(GLIBC_2.2) [SUSv3]	pthread_rwlock_init(GLIBC_2.2) [SUSv3]	pthread_rwlock_rdlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_timedrdlock(GLIBC_2.2) [SUSv3]
pthread_rwlock_timedwrlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_tryrdlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_trywrlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_unlock(GLIBC_2.2) [SUSv3]
pthread_rwlock_wrlock(GLIBC_2.2) [SUSv3]	pthread_rwlockattr_destroy(GLIBC_2.2) [SUSv3]	pthread_rwlockattr_getpshared(GLIBC_2.2) [SUSv3]	pthread_rwlockattr_init(GLIBC_2.2) [SUSv3]
pthread_rwlockattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_self(GLIBC_2.2) [SUSv3]	pthread_setcancelstate(GLIBC_2.2) [SUSv3]	pthread_setcanceltype(GLIBC_2.2) [SUSv3]
pthread_setconcurrency(GLIBC_2.2) [SUSv3]	pthread_setspecific(GLIBC_2.2) [SUSv3]	pthread_sigmask(GLIBC_2.2) [SUSv3]	pthread_testcancel(GLIBC_2.2) [SUSv3]
sem_close(GLIBC_2.2) [SUSv3]	sem_destroy(GLIBC_2.2) [SUSv3]	sem_getvalue(GLIBC_2.2) [SUSv3]	sem_init(GLIBC_2.2) [SUSv3]
sem_open(GLIBC_2.2) [SUSv3]	sem_post(GLIBC_2.2) [SUSv3]	sem_timedwait(GLIBC_2.2) [SUSv3]	sem_trywait(GLIBC_2.2) [SUSv3]
sem_unlink(GLIBC_2.2) [SUSv3]	sem_wait(GLIBC_2.2) [SUSv3]		

11.6.4 Thread aware versions of libc interfaces

11.6.4.1 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) [1]~~	~~pread(GLIBC_2.2) [2]~~	~~pwrite(GLIBC_2.2) [2]~~		
~~open64(GLIBC_2.2) [1]~~	~~pread64(GLIBC_2.2) [1]~~	~~pwrite64(GLIBC_2.2) [1]~~		

~~*Referenced Specification(s)*~~

~~**[1].**~~

lseek64(GLIBC_2.	open64(GLIBC_2.	pread(GLIBC_2.2)	pread64(GLIBC_2.

2) [LFS]	2) [LFS]	[SUSv3]	2) [LFS]
pwrite(GLIBC_2.2) [SUSv3]	pwrite64(GLIBC_2.2) [LFS]		

11.7 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 ~~Large File Support~~ISO C (1999)

~~[2].~~ 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.

11.7.1 pthread.h

```

extern void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *,
int);
extern void _pthread_cleanup_push(struct _pthread_cleanup_buffer *,
                                  void (*__routine) (void *)
                                  , void *);
extern int pthread_attr_destroy(pthread_attr_t *);
extern int pthread_attr_getdetachstate(const typedef struct {
                                       int __detachstate;
                                       int __schedpolicy;
                                       struct sched_param
__schedparam;
                                       int __inheritsched;
                                       int __scope;
                                       size_t __guardsize;
                                       int __stackaddr_set;
                                       void *__stackaddr;
                                       unsigned long int __stacksize;}
                                       pthread_attr_t *, int *);
extern int pthread_attr_getinheritsched(const typedef struct {
                                        int __detachstate;
                                        int __schedpolicy;
                                        struct sched_param
__schedparam;
                                        int __inheritsched;
                                        int __scope;
                                        size_t __guardsize;
                                        int __stackaddr_set;
                                        void *__stackaddr;
                                        unsigned long int
__stacksize;}
                                        pthread_attr_t *, int *);
extern int pthread_attr_getschedparam(const typedef struct {
```

```
                                            int __detachstate;
                                            int __schedpolicy;
                                            struct sched_param
__schedparam;
                                            int __inheritsched;
                                            int __scope;
                                            size_t __guardsize;
                                            int __stackaddr_set;
                                            void *__stackaddr;
                                            unsigned long int __stacksize;}
                                            pthread_attr_t *, struct
sched_param {
                                            int sched_priority;}

                                            *);
extern int pthread_attr_getschedpolicy(const typedef struct {
                                             int __detachstate;
                                             int __schedpolicy;
                                             struct sched_param
__schedparam;
                                             int __inheritsched;
                                             int __scope;
                                             size_t __guardsize;
                                             int __stackaddr_set;
                                             void *__stackaddr;
                                             unsigned long int __stacksize;}
                                             pthread_attr_t *, int *);
extern int pthread_attr_getscope(const typedef struct {
                                 int __detachstate;
                                 int __schedpolicy;
                                 struct sched_param __schedparam;
                                 int __inheritsched;
                                 int __scope;
                                 size_t __guardsize;
                                 int __stackaddr_set;
                                 void *__stackaddr;
                                 unsigned long int __stacksize;}
                                 pthread_attr_t *, int *);
extern int pthread_attr_init(pthread_attr_t *);
extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
extern int pthread_attr_setschedparam(pthread_attr_t *, const struct
sched_param {
                                      int sched_priority;}

                                      *);
extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
extern int pthread_attr_setscope(pthread_attr_t *, int);
extern int pthread_cancel(typedef unsigned long int pthread_t);
extern int pthread_cond_broadcast(pthread_cond_t *);
extern int pthread_cond_destroy(pthread_cond_t *);
extern int pthread_cond_init(pthread_cond_t *, const typedef struct {
                             int __dummy;}

                             pthread_condattr_t *);
extern int pthread_cond_signal(pthread_cond_t *);
extern int pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
const struct timespec {
                                  time_t tv_sec; long int tv_nsec;}

                                  *);
extern int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
extern int pthread_condattr_destroy(pthread_condattr_t *);
extern int pthread_condattr_init(pthread_condattr_t *);
extern int pthread_create(pthread_t *, const typedef struct {
                          int __detachstate;
                          int __schedpolicy;
                          struct sched_param __schedparam;
                          int __inheritsched;
                          int __scope;
                          size_t __guardsize;
                          int __stackaddr_set;
                          void *__stackaddr;
                          unsigned long int __stacksize;}
                          pthread_attr_t *,
                          void *(*__start_routine) (void *p1)
                          , void *);
extern int pthread_detach(typedef unsigned long int pthread_t);
extern int pthread_equal(typedef unsigned long int pthread_t,
                         typedef unsigned long int pthread_t);
extern void pthread_exit(void *);
extern int pthread_getschedparam(typedef unsigned long int pthread_t,
                                 int *, struct sched_param {
                                 int sched_priority;}

                                 *);
extern void *pthread_getspecific(typedef unsigned int pthread_key_t);
extern int pthread_join(typedef unsigned long int pthread_t, void **);
extern int pthread_key_create(pthread_key_t *, void (*destr_func) (void
*)
    );
extern int pthread_key_delete(typedef unsigned int pthread_key_t);
extern int pthread_mutex_destroy(pthread_mutex_t *);
extern int pthread_mutex_init(pthread_mutex_t *, const typedef struct
{
                              int __mutexkind;}

                              pthread_mutexattr_t *);
extern int pthread_mutex_lock(pthread_mutex_t *);
extern int pthread_mutex_trylock(pthread_mutex_t *);
extern int pthread_mutex_unlock(pthread_mutex_t *);
extern int pthread_mutexattr_destroy(pthread_mutexattr_t *);
extern int pthread_mutexattr_init(pthread_mutexattr_t *);
extern int pthread_once(pthread_once_t *, void (*init_routine) (void)
    );
extern int pthread_rwlock_destroy(pthread_rwlock_t *);
extern int pthread_rwlock_init(pthread_rwlock_t *,
pthread_rwlockattr_t *);
extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *);
extern int pthread_rwlock_trywrlock(pthread_rwlock_t *);
extern int pthread_rwlock_unlock(pthread_rwlock_t *);
extern int pthread_rwlock_wrlock(pthread_rwlock_t *);
extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
extern int pthread_rwlockattr_getpshared(const typedef struct {
                                         int __lockkind; int
__pshared;}
                                         pthread_rwlockattr_t *, int
*);
extern int pthread_rwlockattr_init(pthread_rwlockattr_t *);
extern int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
extern typedef unsigned long int pthread_t pthread_self(void);
extern int pthread_setcancelstate(int, int *);
extern int pthread_setcanceltype(int, int *);
extern int pthread_setschedparam(typedef unsigned long int pthread_t,
int, const struct sched_param {
                                 int sched_priority;}
                                 *);
extern int pthread_setspecific(typedef unsigned int pthread_key_t,
                               const void *);
extern void pthread_testcancel(void);
extern int pthread_attr_getguardsize(const typedef struct {
                                     int __detachstate;
                                     int __schedpolicy;
                                     struct sched_param __schedparam;
                                     int __inheritsched;
                                     int __scope;
                                     size_t __guardsize;
                                     int __stackaddr_set;
                                     void *__stackaddr;
                                     unsigned long int __stacksize;}
                                     pthread_attr_t *, size_t *);
extern int pthread_attr_setguardsize(pthread_attr_t *,
                                     typedef unsigned long int
size_t);
extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
extern int pthread_attr_getstackaddr(const typedef struct {
                                     int __detachstate;
                                     int __schedpolicy;
                                     struct sched_param __schedparam;
                                     int __inheritsched;
                                     int __scope;
                                     size_t __guardsize;
                                     int __stackaddr_set;
                                     void *__stackaddr;
                                     unsigned long int __stacksize;}
                                     pthread_attr_t *, void **);
extern int pthread_attr_setstacksize(pthread_attr_t *,
                                     typedef unsigned long int
size_t);
extern int pthread_attr_getstacksize(const typedef struct {
                                     int __detachstate;
                                     int __schedpolicy;
                                     struct sched_param __schedparam;
                                     int __inheritsched;
                                     int __scope;
                                     size_t __guardsize;
                                     int __stackaddr_set;
                                     void *__stackaddr;
                                     unsigned long int __stacksize;}
                                     pthread_attr_t *, size_t *);
extern int pthread_mutexattr_gettype(const typedef struct {
                                     int __mutexkind;}
                                     pthread_mutexattr_t *, int *);
extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
extern int pthread_getconcurrency(void);
extern int pthread_setconcurrency(int);
extern int pthread_attr_getstack(const typedef struct {
                                 int __detachstate;
                                 int __schedpolicy;
                                 struct sched_param __schedparam;
                                 int __inheritsched;
                                 int __scope;
                                 size_t __guardsize;
                                 int __stackaddr_set;
                                 void *__stackaddr;
                                 unsigned long int __stacksize;}
                                 pthread_attr_t *, void **, size_t *);
extern int pthread_attr_setstack(pthread_attr_t *, void *,
                                 typedef unsigned long int size_t);
extern int pthread_condattr_getpshared(const typedef struct {
```

```
                                       int __dummy;}
                                       pthread_condattr_t *, int *);
extern int pthread_condattr_setpshared(pthread_condattr_t *, int);
extern int pthread_mutexattr_getpshared(const typedef struct {
                                        int __mutexkind;}
                                        pthread_mutexattr_t *, int *);
extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *, const struct
timespec {
                                      time_t tv_sec; long int
tv_nsec;}

                                      *);
extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *, const struct
timespec {
                                      time_t tv_sec; long int
tv_nsec;}

                                      *);
extern int __register_atfork(void (*prepare) (void)
                             , void (*parent) (void)
                             , void (*child) (void)
                             , void *);
extern int pthread_setschedprio(typedef unsigned long int pthread_t,
int);
```

11.7.2 semaphore.h

```

extern int sem_close(sem_t *);
extern int sem_destroy(sem_t *);
extern int sem_getvalue(sem_t *, int *);
extern int sem_init(sem_t *, int, unsigned int);
extern sem_t *sem_open(const char *, int, ...);
extern int sem_post(sem_t *);
extern int sem_trywait(sem_t *);
extern int sem_unlink(const char *);
extern int sem_wait(sem_t *);
extern int sem_timedwait(sem_t *, const struct timespec *);
```

11.8 Interfaces for libgcc_s

~~ISO POSIX (2003)~~Table 11-31

~~11.7 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] ~~this specification~~This Specification

11.~~7~~8.1 Unwind Library

11.~~7~~8.1.1 Interfaces for Unwind Library

An LSB conforming implementation shall provide the architecture specific functions for Unwind Library specified in Table 11-32, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-32 libgcc_s - Unwind Library Function Interfaces

~~_Unwind_Backtrace(GCC_3.3) [1]~~	~~_Unwind_ForcedUnwind(GCC_3.0) [1]~~	~~_Unwind_GetGR(GCC_3.0) [1]~~	~~_Unwind_GetRegionStart(GCC_3.0) [1]~~	~~_Unwind_Resume_or_Rethrow(GCC_3.3) [1]~~
~~_Unwind_DeleteException(GCC_3.0) [1]~~	~~_Unwind_GetBSP(GCC_3.3.2) [1]~~	~~_Unwind_GetIP(GCC_3.0) [1]~~	~~_Unwind_RaiseException(GCC_3.0) [1]~~	~~_Unwind_SetGR(GCC_3.0) [1]~~
~~_Unwind_FindEnclosingFunction(GCC_3.3) [1]~~	~~_Unwind_GetCFA(GCC_3.3) [1]~~	~~_Unwind_GetLanguageSpecificData(GCC_3.0) [1]~~	~~_Unwind_Resume(GCC_3.0) [1]~~	~~_Unwind_SetIP(GCC_3.0) [1]~~

~~*Referenced Specification(s)*~~

~~**[1].**~~

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

11.9 Data Definitions for libgcc_s

This section defines global identifiers and their values that are associated with interfaces contained in libgcc_s. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

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

This specification uses the ~~this specification~~ISO C (1999)

~~11.8 Interface Definitions for libgcc_s~~

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

~~Other interfaces listed above for libgcc_s shall behave as described in the referenced base document.~~

~~11.9 Interfaces for libdl~~

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.

11.9.1 unwind.h

```

extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                        _Unwind_Stop_Fn, void *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
_Unwind_Context
                                                   *);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct
_Unwind_Exception
                                                  *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                        _Unwind_Stop_Fn, void *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
_Unwind_Context
                                                   *);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct
_Unwind_Exception
                                                  *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                        _Unwind_Stop_Fn, void *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
_Unwind_Context
                                                   *);
```

```
        extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
        extern _Unwind_Reason_Code _Unwind_RaiseException(struct
        _Unwind_Exception
                                                          *);
        extern void _Unwind_Resume(struct _Unwind_Exception *);
        extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
        extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
        extern void _Unwind_DeleteException(struct _Unwind_Exception *);
        extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
        extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                                _Unwind_Stop_Fn, void *);
        extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
        extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
        extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
        _Unwind_Context
                                                           *);
        extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
        extern _Unwind_Reason_Code _Unwind_RaiseException(struct
        _Unwind_Exception
                                                          *);
        extern void _Unwind_Resume(struct _Unwind_Exception *);
        extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
        extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
        extern void _Unwind_DeleteException(struct _Unwind_Exception *);
        extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
        extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                                _Unwind_Stop_Fn, void *);
        extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
        extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
        extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
        _Unwind_Context
                                                           *);
        extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
        extern _Unwind_Reason_Code _Unwind_RaiseException(struct
        _Unwind_Exception
                                                          *);
        extern void _Unwind_Resume(struct _Unwind_Exception *);
        extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
        extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
        extern void _Unwind_DeleteException(struct _Unwind_Exception *);
        extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
        extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                                _Unwind_Stop_Fn, void *);
        extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
        extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
        extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
        extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
        extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
        extern _Unwind_Reason_Code _Unwind_RaiseException(struct
        _Unwind_Exception
                                                          *);
        extern void _Unwind_Resume(struct _Unwind_Exception *);
        extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
        extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
        extern void _Unwind_DeleteException(struct _Unwind_Exception *);
        extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
        extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
                                                _Unwind_Stop_Fn, void *);
```

```
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct
_Unwind_Exception
                                                *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
*);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);
```

11.10 Interface Definitions for libgcc_s

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

Other interfaces listed in ~~Table 11-33~~Section 11.8 shall behave as described in the referenced base document.

_Unwind_DeleteException

Name

`_Unwind_DeleteException` — private C++ error handling method

Synopsis

```
void _Unwind_DeleteException(struct _Unwind_Exception * object);
```

Description

`_Unwind_DeleteException()` deletes the given exception *`object`*. If a given runtime resumes normal execution after catching a foreign exception, it will not know how to delete that exception. Such an exception shall be deleted by calling `_Unwind_DeleteException()`. This is a convenience function that calls the function pointed to by the *exception_cleanup* field of the exception header.

_Unwind_ForcedUnwind

Name

`_Unwind_ForcedUnwind` — private C++ error handling method

Synopsis

```
_Unwind_Reason_Code _Unwind_ForcedUnwind(struct _Unwind_Exception *
object, _Unwind_Stop_Fn stop, void * stop_parameter);
```

Description

`_Unwind_ForcedUnwind()` raises an exception for forced unwinding, passing along the given exception *`object`*, which should have its *exception_class* and *exception_cleanup* fields set. The exception *`object`* has been allocated by the language-specific runtime, and has a language-specific format, except that it shall contain an `_Unwind_Exception` struct.

Forced unwinding is a single-phase process. *`stop`* and *`stop_parameter`* control the termination of the unwind process instead of the usual personality routine query. *`stop`* is called for each unwind frame, with the parameteres described for the usual personality routine below, plus an additional *`stop_parameter`*.

Return Value

When *`stop`* identifies the destination frame, it transfers control to the user code as appropriate without returning, normally after calling `_Unwind_DeleteException()`. If not, then it should return an `_Unwind_Reason_Code` value.

If *`stop`* returns any reason code other than `_URC_NO_REASON`, then the stack state is indeterminate from the point of view of the caller of `_Unwind_ForcedUnwind()`. Rather than attempt to return, therefore, the unwind library should use the *`exception_cleanup`* entry in the exception, and then call `abort()`.

_URC_NO_REASON

> This is not the destination from. The unwind runtime will call frame's personality routine with the `_UA_FORCE_UNWIND` and `_UA_CLEANUP_PHASE` flag set in *actions*, and then unwind to the next frame and call the `stop()` function again.

_URC_END_OF_STACK

> In order to allow `_Unwind_ForcedUnwind()` to perform special processing when it reaches the end of the stack, the unwind runtime will call it after the last frame is rejected, with a `NULL` stack pointer in the context, and the `stop()` function shall catch this condition. It may return this code if it cannot handle end-of-stack.

_URC_FATAL_PHASE2_ERROR

> The `stop()` function may return this code for other fatal conditions like stack corruption.

_Unwind_GetGR

Name

`_Unwind_GetGR` — private C++ error handling method

Synopsis

```
_Unwind_Word _Unwind_GetGR(struct _Unwind_Context * context, int index);
```

Description

`_Unwind_GetGR()` returns data at *`index`* found in *`context`*. The register is identified by its index: `0` to `31` are for the fixed registers, and `32` to `127` are for the stacked registers.

During the two phases of unwinding, only GR1 has a guaranteed value, which is the global pointer of the frame referenced by the unwind *`context`*. If the register has its NAT bit set, the behavior is unspecified.

_Unwind_GetIP

Name

`_Unwind_GetIP` — private C++ error handling method

Synopsis

```
_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);
```

Description

`_Unwind_GetIP()` returns the instruction pointer value for the routine identified by the unwind *`context`*.

_Unwind_GetLanguageSpecificData

Name

`_Unwind_GetLanguageSpecificData` — private C++ error handling method

Synopsis

```
_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *
context, uint value);
```

Description

`_Unwind_GetLanguageSpecificData()` returns the address of the language specific data area for the current stack frame.

_Unwind_GetRegionStart

Name

`_Unwind_GetRegionStart` — private C++ error handling method

Synopsis

```
_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);
```

Description

`_Unwind_GetRegionStart()` routine returns the address (i.e., `0`) of the beginning of the procedure or code fragment described by the current unwind descriptor block.

_Unwind_RaiseException

Name

`_Unwind_RaiseException` — private C++ error handling method

Synopsis

```
_Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *
object);
```

Description

`_Unwind_RaiseException()` raises an exception, passing along the given exception *`object`*, which should have its *`exception_class`* and *`exception_cleanup`* fields set. The exception object has been allocated by the language-specific runtime, and has a language-specific format, exception that it shall contain an `_Unwind_Exception`.

Return Value

`_Unwind_RaiseException()` does not return unless an error condition is found. If an error condition occurs, an `_Unwind_Reason_Code` is returnd:

_URC_END_OF_STACK

The unwinder encountered the end of the stack during phase one without finding a handler. The unwind runtime will not have modified the stack. The C++ runtime will normally call `uncaught_exception()` in this case.

_URC_FATAL_PHASE1_ERROR

The unwinder encountered an unexpected error during phase one, because of something like stack corruption. The unwind runtime will not have modified the stack. The C++ runtime will normally call `terminate()` in this case.

_URC_FATAL_PHASE2_ERROR

The unwinder encountered an unexpected error during phase two. This is usually a *throw*, which will call `terminate()`.

_Unwind_Resume

Name

`_Unwind_Resume` — private C++ error handling method

Synopsis

```
void _Unwind_Resume(struct _Unwind_Exception * object);
```

Description

`_Unwind_Resume()` resumes propagation of an existing exception *`object`*. A call to this routine is inserted as the end of a landing pad that performs cleanup, but does not resume normal execution. It causes unwinding to proceed further.

_Unwind_SetGR

Name

`_Unwind_SetGR` — private C++ error handling method

Synopsis

```
void _Unwind_SetGR(struct _Unwind_Context * context, int index, uint value);
```

Description

`_Unwind_SetGR()` sets the *`value`* of the register *`index`*ed for the routine identified by the unwind *`context`*.

_Unwind_SetIP

Name

`_Unwind_SetIP` — private C++ error handling method

Synopsis

```
void _Unwind_SetIP(struct _Unwind_Context * context, uint value);
```

Description

`_Unwind_SetIP()` sets the *`value`* of the instruction pointer for the routine identified by the unwind *`context`*

11.11 Interfaces for libdl

Table 11-33 defines the library name and shared object name for the libdl library

Table 11-33 libdl Definition

Library:	libdl
SONAME:	libdl.so.2

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

[LSB] ~~this specification~~This Specification
[SUSv3] ISO POSIX (2003)

11.~~9.~~11.1 Dynamic Loader

11.~~9~~11.1.1 Interfaces for Dynamic Loader

An LSB conforming implementation shall provide the architecture specific functions for Dynamic Loader specified in Table 11-34, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-34 libdl - Dynamic Loader Function Interfaces

~~dladdr(GLIBC_2.0) [1]~~	~~dlclose(GLIBC_2.0) [2]~~	~~dlerror(GLIBC_2.0) [2]~~	~~dlopen(GLIBC_2.1) [1]~~	~~dlsym(GLIBC_2.0) [1]~~

~~*Referenced Specification(s)*~~

~~[1].~~

dladdr(GLIBC_2.0) [LSB]	dlclose(GLIBC_2.0) [SUSv3]	dlerror(GLIBC_2.0) [SUSv3]	dlopen(GLIBC_2.1) [LSB]
dlsym(GLIBC_2.0) [LSB]			

11.12 Data Definitions for libdl

This section defines global identifiers and their values that are associated with interfaces contained in libdl. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

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

This specification uses the ~~this specification~~ISO C (1999)

~~[2].~~ 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.

11.12.1 dlfcn.h

```

extern int dladdr(const void *, Dl_info *);
extern int dlclose(void *);
extern char *dlerror(void);
extern void *dlopen(char *, int);
extern void *dlsym(void *, char *);
```

11.13 Interfaces for libcrypt

~~ISO POSIX (2003)~~Table 11-35

~~11.10 Interfaces for libcrypt~~

~~Table 11-35~~ defines the library name and shared object name for the libcrypt library

Table 11-35 libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

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

[SUSv3] ISO POSIX (2003)

11.~~10~~13.1 Encryption

11.~~10~~13.1.1 Interfaces for Encryption

An LSB conforming implementation shall provide the architecture specific functions for Encryption specified in Table 11-36, with the full mandatory functionality as described in the referenced underlying specification.

Table 11-36 libcrypt - Encryption Function Interfaces

~~crypt(GLIBC_2.0) [1]~~	~~encrypt(GLIBC_2.0) [1]~~	~~setkey(GLIBC_2.0) [1]~~		

~~*Referenced Specification(s)*~~

~~**[1].** ISO POSIX (2003)~~

crypt(GLIBC_2.0) [SUSv3]	encrypt(GLIBC_2.0) [SUSv3]	setkey(GLIBC_2.0) [SUSv3]	

1

IV Utility Libraries

12 Libraries

An LSB-conforming implementation shall also support some utility libraries which are built on top of the interfaces provided by the base libraries. These libraries implement common functionality, and hide additional system dependent information such as file formats and device names.

12.1 Interfaces for libz

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

Table 12-1 libz Definition

Library:	libz
SONAME:	libz.so.1

12.1.1 Compression Library

12.1.1.1 Interfaces for Compression Library

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

12.2 Data Definitions for libz

This section defines global identifiers and their values that are associated with interfaces contained in libz. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

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

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C . 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.

12.2.1 zlib.h

```

extern int gzread(gzFile, voidp, unsigned int);
extern int gzclose(gzFile);
extern gzFile gzopen(const char *, const char *);
extern gzFile gzdopen(int, const char *);
extern int gzwrite(gzFile, voidpc, unsigned int);
extern int gzflush(gzFile, int);
extern const char *gzerror(gzFile, int *);
extern uLong adler32(uLong, const Bytef *, uInt);
extern int compress(Bytef *, uLongf *, const Bytef *, uLong);
extern int compress2(Bytef *, uLongf *, const Bytef *, uLong, int);
extern uLong crc32(uLong, const Bytef *, uInt);
extern int deflate(z_streamp, int);
extern int deflateCopy(z_streamp, z_streamp);
extern int deflateEnd(z_streamp);
extern int deflateInit2_(z_streamp, int, int, int, int, int, const char
*,
                         int);
extern int deflateInit_(z_streamp, int, const char *, int);
extern int deflateParams(z_streamp, int, int);
extern int deflateReset(z_streamp);
extern int deflateSetDictionary(z_streamp, const Bytef *, uInt);
extern const uLongf *get_crc_table(void);
extern int gzeof(gzFile);
extern int gzgetc(gzFile);
extern char *gzgets(gzFile, char *, int);
extern int gzprintf(gzFile, const char *, ...);
extern int gzputc(gzFile, int);
extern int gzputs(gzFile, const char *);
extern int gzrewind(gzFile);
extern z_off_t gzseek(gzFile, z_off_t, int);
extern int gzsetparams(gzFile, int, int);
extern z_off_t gztell(gzFile);
extern int inflate(z_streamp, int);
extern int inflateEnd(z_streamp);
extern int inflateInit2_(z_streamp, int, const char *, int);
extern int inflateInit_(z_streamp, const char *, int);
extern int inflateReset(z_streamp);
extern int inflateSetDictionary(z_streamp, const Bytef *, uInt);
extern int inflateSync(z_streamp);
extern int inflateSyncPoint(z_streamp);
extern int uncompress(Bytef *, uLongf *, const Bytef *, uLong);
extern const char *zError(int);
extern const char *zlibVersion(void);
extern uLong deflateBound(z_streamp, uLong);
extern uLong compressBound(uLong);
```

12.3 Interfaces for libncurses

Table 12-2 defines the library name and shared object name for the libncurses library

Table 12-2 libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

12.~~2~~3.1 Curses

12.~~2~~3.1.1 Interfaces for Curses

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

12.~~3~~4 Data Definitions for libncurses

This section defines global identifiers and their values that are associated with interfaces contained in libncurses. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

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

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C . 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.

12.4.1 curses.h

```

extern int addch(const chtype);
extern int addchnstr(const chtype *, int);
extern int addchstr(const chtype *);
extern int addnstr(const char *, int);
extern int addstr(const char *);
extern int attroff(int);
extern int attron(int);
extern int attrset(int);
extern int attr_get(attr_t *, short *, void *);
extern int attr_off(attr_t, void *);
extern int attr_on(attr_t, void *);
extern int attr_set(attr_t, short, void *);
extern int baudrate(void);
extern int beep(void);
extern int bkgd(chtype);
extern void bkgdset(chtype);
extern int border(chtype, chtype, chtype, chtype, chtype, chtype,
chtype,
                  chtype);
extern int box(WINDOW *, chtype, chtype);
extern bool can_change_color(void);
extern int cbreak(void);
extern int chgat(int, attr_t, short, const void *);
extern int clear(void);
extern int clearok(WINDOW *, bool);
extern int clrtobot(void);
extern int clrtoeol(void);
extern int color_content(short, short *, short *, short *);
extern int color_set(short, void *);
extern int copywin(const WINDOW *, WINDOW *, int, int, int, int, int,
int,
                   int);
extern int curs_set(int);
extern int def_prog_mode(void);
extern int def_shell_mode(void);
extern int delay_output(int);
extern int delch(void);
extern void delscreen(SCREEN *);
extern int delwin(WINDOW *);
extern int deleteln(void);
extern WINDOW *derwin(WINDOW *, int, int, int, int);
extern int doupdate(void);
extern WINDOW *dupwin(WINDOW *);
extern int echo(void);
extern int echochar(const chtype);
extern int erase(void);
extern int endwin(void);
extern char erasechar(void);
extern void filter(void);
extern int flash(void);
```

```
          extern int flushinp(void);
          extern chtype getbkgd(WINDOW *);
          extern int getch(void);
          extern int getnstr(char *, int);
          extern int getstr(char *);
          extern WINDOW *getwin(FILE *);
          extern int halfdelay(int);
          extern bool has_colors(void);
          extern bool has_ic(void);
          extern bool has_il(void);
          extern int hline(chtype, int);
          extern void idcok(WINDOW *, bool);
          extern int idlok(WINDOW *, bool);
          extern void immedok(WINDOW *, bool);
          extern chtype inch(void);
          extern int inchnstr(chtype *, int);
          extern int inchstr(chtype *);
          extern WINDOW *initscr(void);
          extern int init_color(short, short, short, short);
          extern int init_pair(short, short, short);
          extern int innstr(char *, int);
          extern int insch(chtype);
          extern int insdelln(int);
          extern int insertln(void);
          extern int insnstr(const char *, int);
          extern int insstr(const char *);
          extern int instr(char *);
          extern int intrflush(WINDOW *, bool);
          extern bool isendwin(void);
          extern bool is_linetouched(WINDOW *, int);
          extern bool is_wintouched(WINDOW *);
          extern const char *keyname(int);
          extern int keypad(WINDOW *, bool);
          extern char killchar(void);
          extern int leaveok(WINDOW *, bool);
          extern char *longname(void);
          extern int meta(WINDOW *, bool);
          extern int move(int, int);
          extern int mvaddch(int, int, const chtype);
          extern int mvaddchnstr(int, int, const chtype *, int);
          extern int mvaddchstr(int, int, const chtype *);
          extern int mvaddnstr(int, int, const char *, int);
          extern int mvaddstr(int, int, const char *);
          extern int mvchgat(int, int, int, attr_t, short, const void *);
          extern int mvcur(int, int, int, int);
          extern int mvdelch(int, int);
          extern int mvderwin(WINDOW *, int, int);
          extern int mvgetch(int, int);
          extern int mvgetnstr(int, int, char *, int);
          extern int mvgetstr(int, int, char *);
          extern int mvhline(int, int, chtype, int);
          extern chtype mvinch(int, int);
          extern int mvinchnstr(int, int, chtype *, int);
          extern int mvinchstr(int, int, chtype *);
          extern int mvinnstr(int, int, char *, int);
          extern int mvinsch(int, int, chtype);
          extern int mvinsnstr(int, int, const char *, int);
          extern int mvinsstr(int, int, const char *);
          extern int mvinstr(int, int, char *);
          extern int mvprintw(int, int, char *, ...);
          extern int mvscanw(int, int, const char *, ...);
          extern int mvvline(int, int, chtype, int);
          extern int mvwaddch(WINDOW *, int, int, const chtype);
          extern int mvwaddchnstr(WINDOW *, int, int, const chtype *, int);
```

```
        extern int mvwaddchstr(WINDOW *, int, int, const chtype *);
        extern int mvwaddnstr(WINDOW *, int, int, const char *, int);
        extern int mvwaddstr(WINDOW *, int, int, const char *);
        extern int mvwchgat(WINDOW *, int, int, int, attr_t, short, const void
        *);
        extern int mvwdelch(WINDOW *, int, int);
        extern int mvwgetch(WINDOW *, int, int);
        extern int mvwgetnstr(WINDOW *, int, int, char *, int);
        extern int mvwgetstr(WINDOW *, int, int, char *);
        extern int mvwhline(WINDOW *, int, int, chtype, int);
        extern int mvwin(WINDOW *, int, int);
        extern chtype mvwinch(WINDOW *, int, int);
        extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
        extern int mvwinchstr(WINDOW *, int, int, chtype *);
        extern int mvwinnstr(WINDOW *, int, int, char *, int);
        extern int mvwinsch(WINDOW *, int, int, chtype);
        extern int mvwinsnstr(WINDOW *, int, int, const char *, int);
        extern int mvwinsstr(WINDOW *, int, int, const char *);
        extern int mvwinstr(WINDOW *, int, int, char *);
        extern int mvwprintw(WINDOW *, int, int, char *, ...);
        extern int mvwscanw(WINDOW *, int, int, const char *, ...);
        extern int mvwvline(WINDOW *, int, int, chtype, int);
        extern int napms(int);
        extern WINDOW *newpad(int, int);
        extern SCREEN *newterm(const char *, FILE *, FILE *);
        extern WINDOW *newwin(int, int, int, int);
        extern int nl(void);
        extern int nocbreak(void);
        extern int nodelay(WINDOW *, bool);
        extern int noecho(void);
        extern int nonl(void);
        extern void noqiflush(void);
        extern int noraw(void);
        extern int notimeout(WINDOW *, bool);
        extern int overlay(const WINDOW *, WINDOW *);
        extern int overwrite(const WINDOW *, WINDOW *);
        extern int pair_content(short, short *, short *);
        extern int pechochar(WINDOW *, chtype);
        extern int pnoutrefresh(WINDOW *, int, int, int, int, int, int);
        extern int prefresh(WINDOW *, int, int, int, int, int, int);
        extern int printw(char *, ...);
        extern int putwin(WINDOW *, FILE *);
        extern void qiflush(void);
        extern int raw(void);
        extern int redrawwin(WINDOW *);
        extern int refresh(void);
        extern int resetty(void);
        extern int reset_prog_mode(void);
        extern int reset_shell_mode(void);
        extern int ripoffline(int, int (*init) (WINDOW *, int)
            );
        extern int savetty(void);
        extern int scanw(const char *, ...);
        extern int scr_dump(const char *);
        extern int scr_init(const char *);
        extern int scrl(int);
        extern int scroll(WINDOW *);
        extern int scrollok(WINDOW *, typedef unsigned char bool);
        extern int scr_restore(const char *);
        extern int scr_set(const char *);
        extern int setscrreg(int, int);
        extern SCREEN *set_term(SCREEN *);
        extern int slk_attroff(const typedef unsigned long int chtype);
        extern int slk_attron(const typedef unsigned long int chtype);
```

```
          extern int slk_attrset(const typedef unsigned long int chtype);
          extern int slk_attr_set(const typedef chtype attr_t, short, void *);
          extern int slk_clear(void);
          extern int slk_color(short);
          extern int slk_init(int);
          extern char *slk_label(int);
          extern int slk_noutrefresh(void);
          extern int slk_refresh(void);
          extern int slk_restore(void);
          extern int slk_set(int, const char *, int);
          extern int slk_touch(void);
          extern int standout(void);
          extern int standend(void);
          extern int start_color(void);
          extern WINDOW *subpad(WINDOW *, int, int, int, int);
          extern WINDOW *subwin(WINDOW *, int, int, int, int);
          extern int syncok(WINDOW *, typedef unsigned char bool);
          extern typedef unsigned long int chtype termattrs(void);
          extern char *termname(void);
          extern void timeout(int);
          extern int typeahead(int);
          extern int ungetch(int);
          extern int untouchwin(WINDOW *);
          extern void use_env(typedef unsigned char bool);
          extern int vidattr(typedef unsigned long int chtype);
          extern int vidputs(typedef unsigned long int chtype,
                             int (*vidputs_int) (int)
              );
          extern int vline(typedef unsigned long int chtype, int);
          extern int vwprintw(WINDOW *, char *, typedef void *va_list);
          extern int vw_printw(WINDOW *, const char *, typedef void *va_list);
          extern int vwscanw(WINDOW *, const char *, typedef void *va_list);
          extern int vw_scanw(WINDOW *, const char *, typedef void *va_list);
          extern int waddch(WINDOW *, const typedef unsigned long int chtype);
          extern int waddchnstr(WINDOW *, const typedef unsigned long int chtype
          *,
                                int);
          extern int waddchstr(WINDOW *, const typedef unsigned long int chtype
          *);
          extern int waddnstr(WINDOW *, const char *, int);
          extern int waddstr(WINDOW *, const char *);
          extern int wattron(WINDOW *, int);
          extern int wattroff(WINDOW *, int);
          extern int wattrset(WINDOW *, int);
          extern int wattr_get(WINDOW *, attr_t *, short *, void *);
          extern int wattr_on(WINDOW *, typedef chtype attr_t, void *);
          extern int wattr_off(WINDOW *, typedef chtype attr_t, void *);
          extern int wattr_set(WINDOW *, typedef chtype attr_t, short, void *);
          extern int wbkgd(WINDOW *, typedef unsigned long int chtype);
          extern void wbkgdset(WINDOW *, typedef unsigned long int chtype);
          extern int wborder(WINDOW *, typedef unsigned long int chtype,
                             typedef unsigned long int chtype,
                             typedef unsigned long int chtype,
                             typedef unsigned long int chtype,
                             typedef unsigned long int chtype,
                             typedef unsigned long int chtype,
                             typedef unsigned long int chtype,
                             typedef unsigned long int chtype);
          extern int wchgat(WINDOW *, int, typedef chtype attr_t, short,
                            const void *);
          extern int wclear(WINDOW *);
          extern int wclrtobot(WINDOW *);
          extern int wclrtoeol(WINDOW *);
          extern int wcolor_set(WINDOW *, short, void *);
```

```
extern void wcursyncup(WINDOW *);
extern int wdelch(WINDOW *);
extern int wdeleteln(WINDOW *);
extern int wechochar(WINDOW *, const typedef unsigned long int chtype);
extern int werase(WINDOW *);
extern int wgetch(WINDOW *);
extern int wgetnstr(WINDOW *, char *, int);
extern int wgetstr(WINDOW *, char *);
extern int whline(WINDOW *, typedef unsigned long int chtype, int);
extern typedef unsigned long int chtype winch(WINDOW *);
extern int winchnstr(WINDOW *, chtype *, int);
extern int winchstr(WINDOW *, chtype *);
extern int winnstr(WINDOW *, char *, int);
extern int winsch(WINDOW *, typedef unsigned long int chtype);
extern int winsdelln(WINDOW *, int);
extern int winsertln(WINDOW *);
extern int winsnstr(WINDOW *, const char *, int);
extern int winsstr(WINDOW *, const char *);
extern int winstr(WINDOW *, char *);
extern int wmove(WINDOW *, int, int);
extern int wnoutrefresh(WINDOW *);
extern int wprintw(WINDOW *, char *, ...);
extern int wredrawln(WINDOW *, int, int);
extern int wrefresh(WINDOW *);
extern int wscanw(WINDOW *, const char *, ...);
extern int wscrl(WINDOW *, int);
extern int wsetscrreg(WINDOW *, int, int);
extern int wstandout(WINDOW *);
extern int wstandend(WINDOW *);
extern void wsyncdown(WINDOW *);
extern void wsyncup(WINDOW *);
extern void wtimeout(WINDOW *, int);
extern int wtouchln(WINDOW *, int, int, int);
extern int wvline(WINDOW *, typedef unsigned long int chtype, int);
extern char *unctrl(typedef unsigned long int chtype);
extern int COLORS(void);
extern int COLOR_PAIRS(void);
extern chtype acs_map(void);
extern WINDOW *curscr(void);
extern WINDOW *stdscr(void);
extern int COLS(void);
extern int LINES(void);
extern int touchline(WINDOW *, int, int);
extern int touchwin(WINDOW *);
```

12.4.2 term.h

```

extern int putp(const char *);
extern int tigetflag(const char *);
extern int tigetnum(const char *);
extern char *tigetstr(const char *);
extern char *tparm(const char *, ...);
extern TERMINAL *set_curterm(TERMINAL *);
extern int del_curterm(TERMINAL *);
extern int restartterm(char *, int, int *);
extern int setupterm(char *, int, int *);
extern char *tgetstr(char *, char **);
extern char *tgoto(const char *, int, int);
extern int tgetent(char *, const char *);
extern int tgetflag(char *);
extern int tgetnum(char *);
extern int tputs(const char *, int, int (*putcproc) (int)
    );
```

```
extern TERMINAL *cur_term(void);
```

12.5 Interfaces for libutil

Table 12-3 defines the library name and shared object name for the libutil library

Table 12-3 libutil Definition

Library:	libutil
SONAME:	libutil.so.1

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

[LSB] ~~this specification~~This Specification

12.~~3~~5.1 Utility Functions

12.~~3~~5.1.1 Interfaces for Utility Functions

An LSB conforming implementation shall provide the architecture specific functions for Utility Functions specified in Table 12-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 12-4 libutil - Utility Functions Function Interfaces

~~forkpty(GLIBC_2.0) [1]~~	~~login_tty(GLIBC_2.0) [1]~~	~~logwtmp(GLIBC_2.0) [1]~~		
~~login(GLIBC_2.0) [1]~~	~~logout(GLIBC_2.0) [1]~~	~~openpty(GLIBC_2.0) [1]~~		

~~*Referenced Specification(s)*~~

~~[1]. this specification~~

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

1

V Package Format and Installation

13 Software Installation

13.1 Package Dependencies

The LSB runtime environment shall provde the following dependencies.

lsb-core-ia64

> This dependency is used to indicate that the application is dependent on features contained in the LSB-Core specification.

These dependencies shall have a version of 3.0.

Other LSB modules may add additional dependencies; such dependencies shall have the format `lsb-`*`module`*`-ia64`.

13.2 Package Architecture Considerations

All packages must specify an architecture of `IA64`. A LSB runtime environment must accept an architecture of `ia64` even if the native architecture is different.

The `archnum` value in the Lead Section shall be 0x0009.

Annex A Alphabetical Listing of Interfaces

A.1 libgcc_s

The behavior of the interfaces in this library is specified by the following Standards.

~~this specification~~This Specification [LSB]

Table A-1 libgcc_s Function Interfaces

~~_Unwind_Backtrace~~_Unwind_Backtrace[~~1]~~LSB]	~~_Unwind_GetCFA~~_Unwind_GetCFA[~~1]~~LSB]	_Unwind_RaiseException[~~1]~~LSB]
_Unwind_DeleteException[~~1]~~LSB]	~~_Unwind_GetGR~~_Unwind_GetGR[~~1]~~LSB]	~~_Unwind_Resume~~_Unwind_Resume[~~1]~~LSB]
_Unwind_FindEnclosingFunction[~~1]~~LSB]	~~_Unwind_GetIP~~_Unwind_GetIP[~~1]~~LSB]	_Unwind_Resume_or_Rethrow[~~1]~~LSB]
~~_Unwind_ForcedUnwind~~_Unwind_ForcedUnwind[~~1]~~LSB]	_Unwind_GetLanguageSpecificData[~~1]~~LSB]	~~_Unwind_SetGR~~_Unwind_SetGR[~~1]~~LSB]
~~_Unwind_GetBSP~~_Unwind_GetBSP[~~1]~~LSB]	_Unwind_GetRegionStart[~~1]~~LSB]	~~_Unwind_SetIP~~_Unwind_SetIP[~~1]~~LSB]

A.2 libm

The behavior of the interfaces in this library is specified by the following Standards.

ISO C (1999) [ISOC99]
ISO POSIX (2003) [SUSv3]

Table A-2 libm Function Interfaces

~~__fpclassifyl~~__fpclassifyl [~~1]~~ISOC99]	~~__signbitl~~__signbitl[~~1]~~ISOC99]	~~exp2l~~exp2l[~~1]~~SUSv3]

Annex B GNU Free Documentation License (Informative)

This specification is published under the terms of the GNU Free Documentation License, Version 1.1, March 2000

B.1 PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

B.2 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

B.3 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

B.4 COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each

B.5 MODIFICATIONS

given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

B.6 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

B.7 COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

B.8 AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

B.9 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

B.10 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

B.11 FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

B.12 How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.