Linux Standard Base Core Specification for IA32 2.0.1

Specification Introduction

Specification Introduction

Table of Contents

List of Tables

Foreword

This is version 2.0.1 of the Linux Standard Base Core Specification for IA32. An implementation of this version of the specification may not claim to be an implementation of the Linux Standard Base unless it has successfully completed the compliance process as defined by the Free Standards Group.

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.

I. Introductory Elements

Chapter 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") describing those parts of the interface that remain constant across all implementations of the LSB, and an architecture-specific specification ("LSB-arch") 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 IA32 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.

Chapter 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.

Table 2-1. Normative References

Name	Title	URL
DWARF Debugging Information Format	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://www.eagercon.com/dwarf/dwarf-2.0.0.pdf
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEEE Std 754-1985	IEEE Standard 754 for Binary Floating-Point Arithmetic	http://www.ieee.org/
Intel® Architecture Software Developer's Manual Volume 3	The IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide	http://developer.intel.com/design/pentium4/manuals/245472.htm
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
ISO POSIX (2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions 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	http://www.unix.org/version3/
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html

Name	**Title**	**URL**
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 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
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 Command 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	http://www.caldera.com/developers

Name	**Title**	**URL**
	2003	/gabi/2003-12-17/contents.html
System V ABI, IA32 Supplement	System V Application Binary Interface - Intel386™ Architecture Processor Supplement, Fourth Edition	http://www.caldera.com/developers/devspecs/abi386-4.pdf
The Intel® Architecture Software Developer's Manual Volume 1	The IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture	http://developer.intel.com/design/pentium4/manuals/245470.htm
The Intel® Architecture Software Developer's Manual Volume 2	The IA-32 Intel® Architecture Software Developer's Manual Volume 2: Instruction Set Reference	http://developer.intel.com/design/pentium4/manuals/245471.htm
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
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

5

Chapter 3. Requirements

3.1. Relevant Libraries

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

- The implementation shall implement fully the architecture described in the hardware manual for the target processor architecture.
- 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 shall satisfy the following requirements:

- Its executable files are either shell scripts or object files in the format defined for the Object File Format system interface.
- Its object files participate in dynamic linking as defined in the Program Loading and Linking System interface.
- It employs 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 stated in the application's documentation.
- It does 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 in turn an LSB conforming application.
  - The use of that interface or data format, as well as its source, is 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 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.

Chapter 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

Chapter 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).

Chapter 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]

refers to the interface named `forkpty` with symbol version `GLIBC_2.0` that is defined in the first of the listed references below the table.

1

ELF Specification

ELF Specification

Table of Contents

List of Tables

I. Low Level System Information

Chapter 1. Machine Interface

1.1. Processor Architecture

The IA32 Architecture is specified by the following documents

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

Only the features of the Intel486 processor instruction set may be assumed to be present. An application is responsible for determining 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 instructions which do not require elevated privileges may be used.

Applications may not make system calls directly. The interfaces in the C library must be used instead.

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 instruction set 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.

1.2. Data Representation

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

1.2.1. Byte Ordering

See Chapter 3 of the System V ABI, IA32 Supplement.

1.2.2. Fundamental Types

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

Table 1-1. Scalar Types

Type	**C**	`sizeof`	**Alignment (bytes)**	**IntelI386 Architecture**
Integral	long long	8	4	signed double word
	signed long long			
	unsigned long long	8	4	unsigned double

Type	**C**	`sizeof`	**Alignment (bytes)**	**IntelI386 Architecture**
				word

21

1.2.3. Aggregates and Unions

22 See Chapter 3 of the System V ABI, IA32 Supplement.

1.2.4. Bit Fields

23 See Chapter 3 of the System V ABI, IA32 Supplement.

Chapter 2. Function Calling Sequence

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

2.1. CPU Registers

See Chapter 3 of the System V ABI, IA32 Supplement.

2.2. Floating Point Registers

See Chapter 3 of the System V ABI, IA32 Supplement.

2.3. Stack Frame

See Chapter 3 of the System V ABI, IA32 Supplement.

2.4. Arguments

2.4.1. Integral/Pointer

See Chapter 3 of the System V ABI, IA32 Supplement.

2.4.2. Floating Point

See Chapter 3 of the System V ABI, IA32 Supplement.

2.4.3. Struct and Union Point

See Chapter 3 of the System V ABI, IA32 Supplement.

2.4.4. Variable Arguments

See Chapter 3 of the System V ABI, IA32 Supplement.

2.5. Return Values

See Chapter 3 of the System V ABI, IA32 Supplement.

2.5.1. Void

See Chapter 3 of the System V ABI, IA32 Supplement.

2.5.2. Integral/Pointer

12 See Chapter 3 of the System V ABI, IA32 Supplement.

2.5.3. Floating Point

13 See Chapter 3 of the System V ABI, IA32 Supplement.

2.5.4. Struct and Union Point

14 See Chapter 3 of the System V ABI, IA32 Supplement.

Chapter 3. Operating System Interface

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

3.1. Virtual Address Space

See Chapter 3 of the System V ABI, IA32 Supplement.

3.1.1. Page Size

See Chapter 3 of the System V ABI, IA32 Supplement.

3.1.2. Virtual Address Assignments

See Chapter 3 of the System V ABI, IA32 Supplement.

3.1.3. Managing the PRocess Stack

See Chapter 3 of the System V ABI, IA32 Supplement.

3.1.4. Coding Guidlines

See Chapter 3 of the System V ABI, IA32 Supplement.

3.2. Processor Execution Mode

See Chapter 3 of the System V ABI, IA32 Supplement.

3.3. Exception Interface

See Chapter 3 of the System V ABI, IA32 Supplement.

3.3.1. Hardware Exception Types

See Chapter 3 of the System V ABI, IA32 Supplement.

3.3.2. Software Trap Types

See Chapter 3 of the System V ABI, IA32 Supplement.

3.4. Signal Delivery

See Chapter 3 of the System V ABI, IA32 Supplement.

3.4.1. Signal Handler Interface

See Chapter 3 of the System V ABI, IA32 Supplement.

Chapter 4. Process Initialization

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

4.1. Special Registers

See Chapter 3 of the System V ABI, IA32 Supplement.

4.2. Process Stack (on entry)

See Chapter 3 of the System V ABI, IA32 Supplement.

4.3. Auxilliary Vectors

See Chapter 3 of the System V ABI, IA32 Supplement.

4.4. Environment

See Chapter 3 of the System V ABI, IA32 Supplement.

Chapter 5. Coding Examples

LSB-conforming applications may implement fundamental operations using the Coding Examples as defined in Chapter 3 of the System V ABI, IA32 Supplement.

5.1. Code Model Overview/Architecture Constraints

See Chapter 3 of the System V ABI, IA32 Supplement.

5.2. Position-Independent Function Prologue

See Chapter 3 of the System V ABI, IA32 Supplement.

5.3. Data Objects

See Chapter 3 of the System V ABI, IA32 Supplement.

5.3.1. Absolute Load & Store

See Chapter 3 of the System V ABI, IA32 Supplement.

5.3.2. Position Relative Load & Store

See Chapter 3 of the System V ABI, IA32 Supplement.

5.4. Function Calls

See Chapter 3 of the System V ABI, IA32 Supplement.

5.4.1. Absolute Direct Function Call

See Chapter 3 of the System V ABI, IA32 Supplement.

5.4.2. Absolute Indirect Function Call

See Chapter 3 of the System V ABI, IA32 Supplement.

5.4.3. Position-Independent Direct Function Call

See Chapter 3 of the System V ABI, IA32 Supplement.

5.4.4. Position-Independent Indirect Function Call

See Chapter 3 of the System V ABI, IA32 Supplement.

5.5. Branching

13 See Chapter 3 of the System V ABI, IA32 Supplement.

5.5.1. Branch Instruction

14 See Chapter 3 of the System V ABI, IA32 Supplement.

5.5.2. Absolute switch() code

15 See Chapter 3 of the System V ABI, IA32 Supplement.

5.5.3. Position-Independent switch() code

16 See Chapter 3 of the System V ABI, IA32 Supplement.

Chapter 6. C Stack Frame

6.1. Variable Argument List

1 See Chapter 3 of the System V ABI, IA32 Supplement.

6.2. Dynamic Allocation of Stack Space

2 See Chapter 3 of the System V ABI, IA32 Supplement.

Chapter 7. Debug Information

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

II. Object Format

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

Chapter 8. ELF Header

8.1. Machine Information

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

8.1.1. File Class

See Chapter 4 of the System V ABI, IA32 Supplement.

8.1.2. Data Encoding

See Chapter 4 of the System V ABI, IA32 Supplement.

8.1.3. OS Identification

See Chapter 4 of the System V ABI, IA32 Supplement.

8.1.4. Processor Identification

See Chapter 4 of the System V ABI, IA32 Supplement.

8.1.5. Processor Specific Flags

See Chapter 4 of the System V ABI, IA32 Supplement.

Chapter 9. Special Sections

See Chapter 4 of the System V ABI, IA32 Supplement.

9.1. Special Sections

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

9.1.1. ELF Special Sections

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

Table 9-1. ELF Special Sections

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

.got

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

.plt

This section holds the procedure linkage table.

9.1.2. Addition Special Sections

The following additional sections are defined here.

Table 9-2. Additional Special Sections

Name	Type	Attributes
.rel.dyn	SHT_REL	SHF_ALLOC

.rel.dyn

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

Chapter 10. Symbol Table

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

Chapter 11. Relocation

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

11.1. Relocation Types

2 See Chapter 4 of the System V ABI, IA32 Supplement.

III. Program Loading and Dynamic Linking

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

Chapter 12. Program Header

1 See Chapter 5 of the System V ABI, IA32 Supplement.

12.1. Types

12.2. Flags

Chapter 13. Program Loading

1 See Chapter 5 of the System V ABI, IA32 Supplement.

Chapter 14. Dynamic Linking

See Chapter 5 of the System V ABI, IA32 Supplement.

14.1. Dynamic Section

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

DT_PLTGOT

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

14.2. Global Offset Table

See Chapter 5 of the System V ABI, IA32 Supplement.

14.3. Shared Object Dependencies

See Chapter 5 of the System V ABI, IA32 Supplement.

14.4. Function Addresses

See Chapter 5 of the System V ABI, IA32 Supplement.

14.5. Procedure Linkage Table

See Chapter 5 of the System V ABI, IA32 Supplement.

14.6. Initialization and Termination Functions

See Chapter 5 of the System V ABI, IA32 Supplement.

Linux Standard Base Specification

Linux Standard Base Specification

Table of Contents

List of Tables

I. Base Libraries

Chapter 1. Libraries

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

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

1.1. Program Interpreter/Dynamic Linker

The LSB specifies the Program Interpreter to be /lib/ld-lsb.so.2.

1.2. Interfaces for libc

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

Table 1-1. libc Definition

Library:	libc
SONAME:	libc.so.6

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

Large File Support
this specification
SUSv2
ISO POSIX (2003)
SVID Issue 3
SVID Issue 4

1.2.1. RPC

1.2.1.1. Interfaces for RPC

An LSB conforming implementation shall provide the architecture specific functions for RPC specified in Table 1-2, with the full functionality as described in the referenced underlying specification.

Table 1-2. libc - RPC Function Interfaces

authnone_create(GLIBC_2.0) [1]	pmap_unset(GLIBC_2.0) [2]	svcerr_weakauth(GLIBC_2.0) [3]	xdr_float(GLIBC_2.0) [3]	xdr_u_char(GLIBC_2.0) [3]
clnt_create(GLIBC_2.0) [1]	setdomainname(GLIBC_2.0) [2]	svctcp_create(GLIBC_2.0) [2]	xdr_free(GLIBC_2.0) [3]	xdr_u_int(GLIBC_2.0) [2]
clnt_pcreateerror(GLIBC_2.0) [1]	svc_getreqset(GLIBC_2.0) [3]	svcudp_create(GLIBC_2.0) [2]	xdr_int(GLIBC_2.0) [3]	xdr_u_long(GLIBC_2.0) [3]

clnt_perrno(GLIBC_2.0) [1]	svc_register(GLIBC_2.0) [2]	xdr_accepted_reply(GLIBC_2.0) [3]	xdr_long(GLIBC_2.0) [3]	xdr_u_short(GLIBC_2.0) [3]
clnt_perror(GLIBC_2.0) [1]	svc_run(GLIBC_2.0) [2]	xdr_array(GLIBC_2.0) [3]	xdr_opaque(GLIBC_2.0) [3]	xdr_union(GLIBC_2.0) [3]
clnt_spcreateerror(GLIBC_2.0) [1]	svc_sendreply(GLIBC_2.0) [2]	xdr_bool(GLIBC_2.0) [3]	xdr_opaque_auth(GLIBC_2.0) [3]	xdr_vector(GLIBC_2.0) [3]
clnt_sperrno(GLIBC_2.0) [1]	svcerr_auth(GLIBC_2.0) [3]	xdr_bytes(GLIBC_2.0) [3]	xdr_pointer(GLIBC_2.0) [3]	xdr_void(GLIBC_2.0) [3]
clnt_sperror(GLIBC_2.0) [1]	svcerr_decode(GLIBC_2.0) [3]	xdr_callhdr(GLIBC_2.0) [3]	xdr_reference(GLIBC_2.0) [3]	xdr_wrapstring(GLIBC_2.0) [3]
getdomainname(GLIBC_2.0) [2]	svcerr_noproc(GLIBC_2.0) [3]	xdr_callmsg(GLIBC_2.0) [3]	xdr_rejected_reply(GLIBC_2.0) [3]	xdrmem_create(GLIBC_2.0) [3]
key_decryptsession(GLIBC_2.1) [3]	svcerr_noprog(GLIBC_2.0) [3]	xdr_char(GLIBC_2.0) [3]	xdr_replymsg(GLIBC_2.0) [3]	xdrrec_create(GLIBC_2.0) [3]
pmap_getport(GLIBC_2.0) [2]	svcerr_progvers(GLIBC_2.0) [3]	xdr_double(GLIBC_2.0) [3]	xdr_short(GLIBC_2.0) [3]	xdrrec_eof(GLIBC_2.0) [3]
pmap_set(GLIBC_2.0) [2]	svcerr_systemerr(GLIBC_2.0) [3]	xdr_enum(GLIBC_2.0) [3]	xdr_string(GLIBC_2.0) [3]	

Referenced Specification(s)

[1]. SVID Issue 4

[2]. this specification

[3]. SVID Issue 3

1.2.2. System Calls

1.2.2.1. Interfaces for System Calls

An LSB conforming implementation shall provide the architecture specific functions for System Calls specified in Table 1-3, with the full functionality as described in the referenced underlying specification.

Table 1-3. libc - System Calls Function Interfaces

__fxstat(GLIBC_2.0) [1]	fchmod(GLIBC_2.0) [2]	getwd(GLIBC_2.0) [2]	read(GLIBC_2.0) [2]	setrlimit(GLIBC_2.2) [2]
__getpgid(GLIBC_2.0) [1]	fchown(GLIBC_2.0) [2]	initgroups(GLIBC_2.0) [1]	readdir(GLIBC_2.0) [2]	setrlimit64(GLIBC_2.1) [3]
__lxstat(GLIBC_2.0) [1]	fcntl(GLIBC_2.0) [1]	ioctl(GLIBC_2.0) [1]	readdir_r(GLIBC_2.0) [2]	setsid(GLIBC_2.0) [2]
__xmknod(GLIBC_2.0) [1]	fdatasync(GLIBC_2.0) [2]	kill(GLIBC_2.0) [1]	readlink(GLIBC_2.0) [2]	setuid(GLIBC_2.0) [2]

__xstat(GLIBC_2.0) [1]	flock(GLIBC_2.0) [1]	killpg(GLIBC_2.0) [2]	readv(GLIBC_2.0) [2]	sleep(GLIBC_2.0) [2]
access(GLIBC_2.0) [2]	fork(GLIBC_2.0) [2]	lchown(GLIBC_2.0) [2]	rename(GLIBC_2.0) [2]	statvfs(GLIBC_2.1) [2]
acct(GLIBC_2.0) [1]	fstatvfs(GLIBC_2.1) [2]	link(GLIBC_2.0) [2]	rmdir(GLIBC_2.0) [2]	stime(GLIBC_2.0) [1]
alarm(GLIBC_2.0) [2]	fsync(GLIBC_2.0) [2]	lockf(GLIBC_2.0) [2]	sbrk(GLIBC_2.0) [4]	symlink(GLIBC_2.0) [2]
brk(GLIBC_2.0) [4]	ftime(GLIBC_2.0) [2]	lseek(GLIBC_2.0) [2]	sched_get_priority_max(GLIBC_2.0) [2]	sync(GLIBC_2.0) [2]
chdir(GLIBC_2.0) [2]	ftruncate(GLIBC_2.0) [2]	mkdir(GLIBC_2.0) [2]	sched_get_priority_min(GLIBC_2.0) [2]	sysconf(GLIBC_2.0) [2]
chmod(GLIBC_2.0) [2]	getcontext(GLIBC_2.1) [2]	mkfifo(GLIBC_2.0) [2]	sched_getparam(GLIBC_2.0) [2]	time(GLIBC_2.0) [2]
chown(GLIBC_2.1) [2]	getegid(GLIBC_2.0) [2]	mlock(GLIBC_2.0) [2]	sched_getscheduler(GLIBC_2.0) [2]	times(GLIBC_2.0) [2]
chroot(GLIBC_2.0) [4]	geteuid(GLIBC_2.0) [2]	mlockall(GLIBC_2.0) [2]	sched_rr_get_interval(GLIBC_2.0) [2]	truncate(GLIBC_2.0) [2]
clock(GLIBC_2.0) [2]	getgid(GLIBC_2.0) [2]	mmap(GLIBC_2.0) [2]	sched_setparam(GLIBC_2.0) [2]	ulimit(GLIBC_2.0) [2]
close(GLIBC_2.0) [2]	getgroups(GLIBC_2.0) [2]	mprotect(GLIBC_2.0) [2]	sched_setscheduler(GLIBC_2.0) [2]	umask(GLIBC_2.0) [2]
closedir(GLIBC_2.0) [2]	getitimer(GLIBC_2.0) [2]	msync(GLIBC_2.0) [2]	sched_yield(GLIBC_2.0) [2]	uname(GLIBC_2.0) [2]
creat(GLIBC_2.0) [1]	getloadavg(GLIBC_2.2) [1]	munlock(GLIBC_2.0) [2]	select(GLIBC_2.0) [2]	unlink(GLIBC_2.0) [1]
dup(GLIBC_2.0) [2]	getpagesize(GLIBC_2.0) [4]	munlockall(GLIBC_2.0) [2]	setcontext(GLIBC_2.0) [2]	utime(GLIBC_2.0) [2]
dup2(GLIBC_2.0) [2]	getpgid(GLIBC_2.0) [2]	munmap(GLIBC_2.0) [2]	setegid(GLIBC_2.0) [2]	utimes(GLIBC_2.0) [2]
execl(GLIBC_2.0) [2]	getpgrp(GLIBC_2.0) [2]	nanosleep(GLIBC_2.0) [2]	seteuid(GLIBC_2.0) [2]	vfork(GLIBC_2.0) [2]
execle(GLIBC_2.0) [2]	getpid(GLIBC_2.0) [2]	nice(GLIBC_2.0) [2]	setgid(GLIBC_2.0) [2]	wait(GLIBC_2.0) [2]
execlp(GLIBC_2.0) [2]	getppid(GLIBC_2.0) [2]	open(GLIBC_2.0) [1]	setitimer(GLIBC_2.0) [2]	wait3(GLIBC_2.0) [1]

execv(GLIBC_2.0) [2]	getpriority(GLIBC_2.0) [2]	opendir(GLIBC_2.0) [2]	setpgid(GLIBC_2.0) [2]	wait4(GLIBC_2.0) [1]
execve(GLIBC_2.0) [2]	getrlimit(GLIBC_2.2) [2]	pathconf(GLIBC_2.0) [2]	setpgrp(GLIBC_2.0) [2]	waitpid(GLIBC_2.0) [1]
execvp(GLIBC_2.0) [2]	getrusage(GLIBC_2.0) [2]	pause(GLIBC_2.0) [2]	setpriority(GLIBC_2.0) [2]	write(GLIBC_2.0) [2]
exit(GLIBC_2.0) [2]	getsid(GLIBC_2.0) [2]	pipe(GLIBC_2.0) [2]	setregid(GLIBC_2.0) [2]	writev(GLIBC_2.0) [2]
fchdir(GLIBC_2.0) [2]	getuid(GLIBC_2.0) [2]	poll(GLIBC_2.0) [2]	setreuid(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. Large File Support

[4]. SUSv2

1.2.3. Standard I/O

1.2.3.1. Interfaces for Standard I/O

An LSB conforming implementation shall provide the architecture specific functions for Standard I/O specified in Table 1-4, with the full functionality as described in the referenced underlying specification.

Table 1-4. libc - Standard I/O Function Interfaces

_IO_feof(GLIBC_2.0) [1]	fgetpos(GLIBC_2.2) [2]	fsetpos(GLIBC_2.2) [2]	putchar(GLIBC_2.0) [2]	sscanf(GLIBC_2.0) [2]
_IO_getc(GLIBC_2.0) [1]	fgets(GLIBC_2.0) [2]	ftell(GLIBC_2.0) [2]	putchar_unlocked(GLIBC_2.0) [2]	telldir(GLIBC_2.0) [2]
_IO_putc(GLIBC_2.0) [1]	fgetwc_unlocked(GLIBC_2.2) [1]	ftello(GLIBC_2.1) [2]	puts(GLIBC_2.0) [2]	tempnam(GLIBC_2.0) [2]
_IO_puts(GLIBC_2.0) [1]	fileno(GLIBC_2.0) [2]	fwrite(GLIBC_2.0) [2]	putw(GLIBC_2.0) [3]	ungetc(GLIBC_2.0) [2]
asprintf(GLIBC_2.0) [1]	flockfile(GLIBC_2.0) [2]	getc(GLIBC_2.0) [2]	remove(GLIBC_2.0) [2]	vasprintf(GLIBC_2.0) [1]
clearerr(GLIBC_2.0) [2]	fopen(GLIBC_2.1) [1]	getc_unlocked(GLIBC_2.0) [2]	rewind(GLIBC_2.0) [2]	vdprintf(GLIBC_2.0) [1]
ctermid(GLIBC_2.0) [2]	fprintf(GLIBC_2.0) [2]	getchar(GLIBC_2.0) [2]	rewinddir(GLIBC_2.0) [2]	vfprintf(GLIBC_2.0) [2]

fclose(GLIBC_2.1) [2]	fputc(GLIBC_2.0) [2]	getchar_unlocked(GLIBC_2.0) [2]	scanf(GLIBC_2.0) [2]	vprintf(GLIBC_2.0) [2]
fdopen(GLIBC_2.1) [2]	fputs(GLIBC_2.0) [2]	getw(GLIBC_2.0) [3]	seekdir(GLIBC_2.0) [2]	vsnprintf(GLIBC_2.0) [2]
feof(GLIBC_2.0) [2]	fread(GLIBC_2.0) [2]	pclose(GLIBC_2.1) [2]	setbuf(GLIBC_2.0) [2]	vsprintf(GLIBC_2.0) [2]
ferror(GLIBC_2.0) [2]	freopen(GLIBC_2.0) [1]	popen(GLIBC_2.1) [2]	setbuffer(GLIBC_2.0) [1]	
fflush(GLIBC_2.0) [2]	fscanf(GLIBC_2.0) [2]	printf(GLIBC_2.0) [2]	setvbuf(GLIBC_2.0) [2]	
fflush_unlocked(GLIBC_2.0) [1]	fseek(GLIBC_2.0) [2]	putc(GLIBC_2.0) [2]	snprintf(GLIBC_2.0) [2]	
fgetc(GLIBC_2.0) [2]	fseeko(GLIBC_2.1) [2]	putc_unlocked(GLIBC_2.0) [2]	sprintf(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. SUSv2

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard I/O specified in Table 1-5, with the full functionality as described in the referenced underlying specification.

Table 1-5. libc - Standard I/O Data Interfaces

stderr(GLIBC_2.0) [1]	stdin(GLIBC_2.0) [1]	stdout(GLIBC_2.0) [1]		

Referenced Specification(s)

[1]. ISO POSIX (2003)

1.2.4. Signal Handling

1.2.4.1. Interfaces for Signal Handling

An LSB conforming implementation shall provide the architecture specific functions for Signal Handling specified in Table 1-6, with the full functionality as described in the referenced underlying specification.

Table 1-6. libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.1) [1]	sigaddset(GLIBC_2.0) [2]	sighold(GLIBC_2.1) [2]	sigpause(GLIBC_2.0) [2]	sigsuspend(GLIBC_2.0) [2]
__libc_current_sigrt	sigaltstack(GLIBC_	sigignore(GLIBC_2	sigpending(GLIBC_	sigtimedwait(GLIB

min(GLIBC_2.1) [1]	2.0) [2]	.1) [2]	2.0) [2]	C_2.1) [2]
__sigsetjmp(GLIBC_2.0) [1]	sigandset(GLIBC_2.0) [1]	siginterrupt(GLIBC_2.0) [2]	sigprocmask(GLIBC_2.0) [2]	sigwait(GLIBC_2.0) [2]
__sysv_signal(GLIBC_2.0) [1]	sigblock(GLIBC_2.0) [1]	sigisemptyset(GLIBC_2.0) [1]	sigqueue(GLIBC_2.1) [2]	sigwaitinfo(GLIBC_2.1) [2]
bsd_signal(GLIBC_2.0) [2]	sigdelset(GLIBC_2.0) [2]	sigismember(GLIBC_2.0) [2]	sigrelse(GLIBC_2.1) [2]	
psignal(GLIBC_2.0) [1]	sigemptyset(GLIBC_2.0) [2]	siglongjmp(GLIBC_2.0) [2]	sigreturn(GLIBC_2.0) [1]	
raise(GLIBC_2.0) [2]	sigfillset(GLIBC_2.0) [2]	signal(GLIBC_2.0) [2]	sigset(GLIBC_2.1) [2]	
sigaction(GLIBC_2.0) [2]	siggetmask(GLIBC_2.0) [1]	sigorset(GLIBC_2.0) [1]	sigstack(GLIBC_2.0) [3]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. SUSv2

An LSB conforming implementation shall provide the architecture specific data interfaces for Signal Handling specified in Table 1-7, with the full functionality as described in the referenced underlying specification.

Table 1-7. libc - Signal Handling Data Interfaces

_sys_siglist(GLIBC_2.3.3) [1]				

Referenced Specification(s)

[1]. this specification

1.2.5. Localization Functions

1.2.5.1. Interfaces for Localization Functions

An LSB conforming implementation shall provide the architecture specific functions for Localization Functions specified in Table 1-8, with the full functionality as described in the referenced underlying specification.

Table 1-8. libc - Localization Functions Function Interfaces

bind_textdomain_codeset(GLIBC_2.2) [1]	catopen(GLIBC_2.0) [2]	dngettext(GLIBC_2.2) [1]	iconv_open(GLIBC_2.1) [2]	setlocale(GLIBC_2.0) [2]
bindtextdomain(GL	dcgettext(GLIBC_2.	gettext(GLIBC_2.0)	localeconv(GLIBC_	textdomain(GLIBC

IBC_2.0) [1]	0) [1]	[1]	2.2) [2]	_2.0) [1]
catclose(GLIBC_2.0) [2]	dcngettext(GLIBC_2.2) [1]	iconv(GLIBC_2.1) [2]	ngettext(GLIBC_2.2) [1]	
catgets(GLIBC_2.0) [2]	dgettext(GLIBC_2.0) [1]	iconv_close(GLIBC_2.1) [2]	nl_langinfo(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

An LSB conforming implementation shall provide the architecture specific data interfaces for Localization Functions specified in Table 1-9, with the full functionality as described in the referenced underlying specification.

Table 1-9. libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr(GLIBC_2.0) [1]				

Referenced Specification(s)

[1]. this specification

1.2.6. Socket Interface

1.2.6.1. Interfaces for Socket Interface

An LSB conforming implementation shall provide the architecture specific functions for Socket Interface specified in Table 1-10, with the full functionality as described in the referenced underlying specification.

Table 1-10. libc - Socket Interface Function Interfaces

__h_errno_location(GLIBC_2.0) [1]	gethostid(GLIBC_2.0) [2]	listen(GLIBC_2.0) [2]	sendmsg(GLIBC_2.0) [2]	socketpair(GLIBC_2.0) [2]
accept(GLIBC_2.0) [2]	gethostname(GLIBC_2.0) [2]	recv(GLIBC_2.0) [2]	sendto(GLIBC_2.0) [2]	
bind(GLIBC_2.0) [2]	getpeername(GLIBC_2.0) [2]	recvfrom(GLIBC_2.0) [2]	setsockopt(GLIBC_2.0) [1]	
bindresvport(GLIBC_2.0) [1]	getsockname(GLIBC_2.0) [2]	recvmsg(GLIBC_2.0) [2]	shutdown(GLIBC_2.0) [2]	
connect(GLIBC_2.0) [2]	getsockopt(GLIBC_2.0) [2]	send(GLIBC_2.0) [2]	socket(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

An LSB conforming implementation shall provide the architecture specific deprecated functions for Socket Interface specified in Table 1-11, with the full functionality as described in the referenced underlying specification.

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

Table 1-11. libc - Socket Interface Deprecated Function Interfaces

gethostbyname_r(GLIBC_2.1.2) [1]				

Referenced Specification(s)

[1]. this specification

1.2.7. Wide Characters

1.2.7.1. Interfaces for Wide Characters

An LSB conforming implementation shall provide the architecture specific functions for Wide Characters specified in Table 1-12, with the full functionality as described in the referenced underlying specification.

Table 1-12. libc - Wide Characters Function Interfaces

__wcstod_internal(GLIBC_2.0) [1]	mbsinit(GLIBC_2.0) [2]	vwscanf(GLIBC_2.2) [2]	wcsnlen(GLIBC_2.1) [1]	wcstoumax(GLIBC_2.1) [2]
__wcstof_internal(GLIBC_2.0) [1]	mbsnrtowcs(GLIBC_2.0) [1]	wcpcpy(GLIBC_2.0) [1]	wcsnrtombs(GLIBC_2.0) [1]	wcstouq(GLIBC_2.0) [1]
__wcstol_internal(GLIBC_2.0) [1]	mbsrtowcs(GLIBC_2.0) [2]	wcpncpy(GLIBC_2.0) [1]	wcspbrk(GLIBC_2.0) [2]	wcswcs(GLIBC_2.1) [2]
__wcstold_internal(GLIBC_2.0) [1]	mbstowcs(GLIBC_2.0) [2]	wcrtomb(GLIBC_2.0) [2]	wcsrchr(GLIBC_2.0) [2]	wcswidth(GLIBC_2.0) [2]
__wcstoul_internal(GLIBC_2.0) [1]	mbtowc(GLIBC_2.0) [2]	wcscasecmp(GLIBC_2.1) [1]	wcsrtombs(GLIBC_2.0) [2]	wcsxfrm(GLIBC_2.0) [2]
btowc(GLIBC_2.0) [2]	putwc(GLIBC_2.2) [2]	wcscat(GLIBC_2.0) [2]	wcsspn(GLIBC_2.0) [2]	wctob(GLIBC_2.0) [2]
fgetwc(GLIBC_2.2) [2]	putwchar(GLIBC_2.2) [2]	wcschr(GLIBC_2.0) [2]	wcsstr(GLIBC_2.0) [2]	wctomb(GLIBC_2.0) [2]
fgetws(GLIBC_2.2) [2]	swprintf(GLIBC_2.2) [2]	wcscmp(GLIBC_2.0) [2]	wcstod(GLIBC_2.0) [2]	wctrans(GLIBC_2.0) [2]
fputwc(GLIBC_2.2) [2]	swscanf(GLIBC_2.2) [2]	wcscoll(GLIBC_2.0) [2]	wcstof(GLIBC_2.0) [2]	wctype(GLIBC_2.0) [2]
fputws(GLIBC_2.2) [2]	towctrans(GLIBC_2.0) [2]	wcscpy(GLIBC_2.0) [2]	wcstoimax(GLIBC_2.1) [2]	wcwidth(GLIBC_2.0) [2]

fwide(GLIBC_2.2) [2]	towlower(GLIBC_2.0) [2]	wcscspn(GLIBC_2.0) [2]	wcstok(GLIBC_2.0) [2]	wmemchr(GLIBC_2.0) [2]
fwprintf(GLIBC_2.2) [2]	towupper(GLIBC_2.0) [2]	wcsdup(GLIBC_2.0) [1]	wcstol(GLIBC_2.0) [2]	wmemcmp(GLIBC_2.0) [2]
fwscanf(GLIBC_2.2) [2]	ungetwc(GLIBC_2.2) [2]	wcsftime(GLIBC_2.2) [2]	wcstold(GLIBC_2.0) [2]	wmemcpy(GLIBC_2.0) [2]
getwc(GLIBC_2.2) [2]	vfwprintf(GLIBC_2.2) [2]	wcslen(GLIBC_2.0) [2]	wcstoll(GLIBC_2.1) [2]	wmemmove(GLIBC_2.0) [2]
getwchar(GLIBC_2.2) [2]	vfwscanf(GLIBC_2.2) [2]	wcsncasecmp(GLIBC_2.1) [1]	wcstombs(GLIBC_2.0) [2]	wmemset(GLIBC_2.0) [2]
mblen(GLIBC_2.0) [2]	vswprintf(GLIBC_2.2) [2]	wcsncat(GLIBC_2.0) [2]	wcstoq(GLIBC_2.0) [1]	wprintf(GLIBC_2.2) [2]
mbrlen(GLIBC_2.0) [2]	vswscanf(GLIBC_2.2) [2]	wcsncmp(GLIBC_2.0) [2]	wcstoul(GLIBC_2.0) [2]	wscanf(GLIBC_2.2) [2]
mbrtowc(GLIBC_2.0) [2]	vwprintf(GLIBC_2.2) [2]	wcsncpy(GLIBC_2.0) [2]	wcstoull(GLIBC_2.1) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

1.2.8. String Functions

1.2.8.1. Interfaces for String Functions

An LSB conforming implementation shall provide the architecture specific functions for String Functions specified in Table 1-13, with the full functionality as described in the referenced underlying specification.

Table 1-13. libc - String Functions Function Interfaces

__mempcpy(GLIBC_2.0) [1]	bzero(GLIBC_2.0) [2]	strcasestr(GLIBC_2.1) [1]	strncasecmp(GLIBC_2.0) [2]	strtoimax(GLIBC_2.1) [2]
__rawmemchr(GLIBC_2.1) [1]	ffs(GLIBC_2.0) [2]	strcat(GLIBC_2.0) [2]	strncat(GLIBC_2.0) [2]	strtok(GLIBC_2.0) [2]
__stpcpy(GLIBC_2.0) [1]	index(GLIBC_2.0) [2]	strchr(GLIBC_2.0) [2]	strncmp(GLIBC_2.0) [2]	strtok_r(GLIBC_2.0) [2]
__strdup(GLIBC_2.0) [1]	memccpy(GLIBC_2.0) [2]	strcmp(GLIBC_2.0) [2]	strncpy(GLIBC_2.0) [2]	strtold(GLIBC_2.0) [2]
__strtod_internal(GLIBC_2.0) [1]	memchr(GLIBC_2.0) [2]	strcoll(GLIBC_2.0) [2]	strndup(GLIBC_2.0) [1]	strtoll(GLIBC_2.0) [2]
__strtof_internal(G	memcmp(GLIBC_2	strcpy(GLIBC_2.0)	strnlen(GLIBC_2.0)	strtoq(GLIBC_2.0)

LIBC_2.0) [1]	.0) [2]	[2]	[1]	[1]
__strtok_r(GLIBC_2.0) [1]	memcpy(GLIBC_2.0) [2]	strcspn(GLIBC_2.0) [2]	strpbrk(GLIBC_2.0) [2]	strtoull(GLIBC_2.0) [2]
__strtol_internal(GLIBC_2.0) [1]	memmove(GLIBC_2.0) [2]	strdup(GLIBC_2.0) [2]	strptime(GLIBC_2.0) [1]	strtoumax(GLIBC_2.1) [2]
__strtold_internal(GLIBC_2.0) [1]	memrchr(GLIBC_2.2) [1]	strerror(GLIBC_2.0) [2]	strrchr(GLIBC_2.0) [2]	strtouq(GLIBC_2.0) [1]
__strtoll_internal(GLIBC_2.0) [1]	memset(GLIBC_2.0) [2]	strerror_r(GLIBC_2.0) [1]	strsep(GLIBC_2.0) [1]	strverscmp(GLIBC_2.1) [1]
__strtoul_internal(GLIBC_2.0) [1]	rindex(GLIBC_2.0) [2]	strfmon(GLIBC_2.0) [2]	strsignal(GLIBC_2.0) [1]	strxfrm(GLIBC_2.0) [2]
__strtoull_internal(GLIBC_2.0) [1]	stpcpy(GLIBC_2.0) [1]	strfry(GLIBC_2.0) [1]	strspn(GLIBC_2.0) [2]	swab(GLIBC_2.0) [2]
bcmp(GLIBC_2.0) [2]	stpncpy(GLIBC_2.0) [1]	strftime(GLIBC_2.0) [2]	strstr(GLIBC_2.0) [2]	
bcopy(GLIBC_2.0) [2]	strcasecmp(GLIBC_2.0) [2]	strlen(GLIBC_2.0) [2]	strtof(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

1.2.9. IPC Functions

1.2.9.1. Interfaces for IPC Functions

An LSB conforming implementation shall provide the architecture specific functions for IPC Functions specified in Table 1-14, with the full functionality as described in the referenced underlying specification.

Table 1-14. libc - IPC Functions Function Interfaces

ftok(GLIBC_2.0) [1]	msgrcv(GLIBC_2.0) [1]	semget(GLIBC_2.0) [1]	shmctl(GLIBC_2.2) [1]	
msgctl(GLIBC_2.2) [1]	msgsnd(GLIBC_2.0) [1]	semop(GLIBC_2.0) [1]	shmdt(GLIBC_2.0) [1]	
msgget(GLIBC_2.0) [1]	semctl(GLIBC_2.2) [1]	shmat(GLIBC_2.0) [1]	shmget(GLIBC_2.0) [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

1.2.10. Regular Expressions

1.2.10.1. Interfaces for Regular Expressions

An LSB conforming implementation shall provide the architecture specific functions for Regular Expressions specified in Table 1-15, with the full functionality as described in the referenced underlying specification.

Table 1-15. libc - Regular Expressions Function Interfaces

regcomp(GLIBC_2.0) [1]	regerror(GLIBC_2.0) [1]	regexec(GLIBC_2.0) [1]	regfree(GLIBC_2.0) [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

An LSB conforming implementation shall provide the architecture specific deprecated functions for Regular Expressions specified in Table 1-16, with the full functionality as described in the referenced underlying specification.

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

Table 1-16. libc - Regular Expressions Deprecated Function Interfaces

advance(GLIBC_2.0) [1]	re_comp(GLIBC_2.0) [1]	re_exec(GLIBC_2.0) [1]	step(GLIBC_2.0) [1]	

Referenced Specification(s)

[1]. SUSv2

An LSB conforming implementation shall provide the architecture specific deprecated data interfaces for Regular Expressions specified in Table 1-17, with the full functionality as described in the referenced underlying specification.

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

Table 1-17. libc - Regular Expressions Deprecated Data Interfaces

loc1(GLIBC_2.0) [1]	loc2(GLIBC_2.0) [1]	locs(GLIBC_2.0) [1]		

Referenced Specification(s)

[1]. SUSv2

1.2.11. Character Type Functions

1.2.11.1. Interfaces for Character Type Functions

An LSB conforming implementation shall provide the architecture specific functions for Character Type Functions specified in Table 1-18, with the full functionality as described in the referenced underlying specification.

Table 1-18. libc - Character Type Functions Function Interfaces

__ctype_get_mb_cur_max(GLIBC_2.0) [1]	isdigit(GLIBC_2.0) [2]	iswalnum(GLIBC_2.0) [2]	iswlower(GLIBC_2.0) [2]	toascii(GLIBC_2.0) [2]
_tolower(GLIBC_2.0) [2]	isgraph(GLIBC_2.0) [2]	iswalpha(GLIBC_2.0) [2]	iswprint(GLIBC_2.0) [2]	tolower(GLIBC_2.0) [2]
_toupper(GLIBC_2.0) [2]	islower(GLIBC_2.0) [2]	iswblank(GLIBC_2.1) [2]	iswpunct(GLIBC_2.0) [2]	toupper(GLIBC_2.0) [2]
isalnum(GLIBC_2.0) [2]	isprint(GLIBC_2.0) [2]	iswcntrl(GLIBC_2.0) [2]	iswspace(GLIBC_2.0) [2]	
isalpha(GLIBC_2.0) [2]	ispunct(GLIBC_2.0) [2]	iswctype(GLIBC_2.0) [2]	iswupper(GLIBC_2.0) [2]	
isascii(GLIBC_2.0) [2]	isspace(GLIBC_2.0) [2]	iswdigit(GLIBC_2.0) [2]	iswxdigit(GLIBC_2.0) [2]	
iscntrl(GLIBC_2.0) [2]	isupper(GLIBC_2.0) [2]	iswgraph(GLIBC_2.0) [2]	isxdigit(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

1.2.12. Time Manipulation

1.2.12.1. Interfaces for Time Manipulation

An LSB conforming implementation shall provide the architecture specific functions for Time Manipulation specified in Table 1-19, with the full functionality as described in the referenced underlying specification.

Table 1-19. libc - Time Manipulation Function Interfaces

adjtime(GLIBC_2.0) [1]	ctime(GLIBC_2.0) [2]	gmtime(GLIBC_2.0) [2]	localtime_r(GLIBC_2.0) [2]	ualarm(GLIBC_2.0) [2]
asctime(GLIBC_2.0) [2]	ctime_r(GLIBC_2.0) [2]	gmtime_r(GLIBC_2.0) [2]	mktime(GLIBC_2.0) [2]	
asctime_r(GLIBC_2.0) [2]	difftime(GLIBC_2.0) [2]	localtime(GLIBC_2.0) [2]	tzset(GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

An LSB conforming implementation shall provide the architecture specific deprecated functions for Time Manipulation specified in Table 1-20, with the full functionality as described in the referenced underlying specification.

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

Table 1-20. libc - Time Manipulation Deprecated Function Interfaces

adjtimex(GLIBC_2.0) [1]				

Referenced Specification(s)

[1]. this specification

An LSB conforming implementation shall provide the architecture specific data interfaces for Time Manipulation specified in Table 1-21, with the full functionality as described in the referenced underlying specification.

Table 1-21. libc - Time Manipulation Data Interfaces

__daylight(GLIBC_2.0) [1]	__tzname(GLIBC_2.0) [1]	timezone(GLIBC_2.0) [2]		
__timezone(GLIBC_2.0) [1]	daylight(GLIBC_2.0) [2]	tzname(GLIBC_2.0) [2]		

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

1.2.13. Terminal Interface Functions

1.2.13.1. Interfaces for Terminal Interface Functions

An LSB conforming implementation shall provide the architecture specific functions for Terminal Interface Functions specified in Table 1-22, with the full functionality as described in the referenced underlying specification.

Table 1-22. libc - Terminal Interface Functions Function Interfaces

cfgetispeed(GLIBC_2.0) [1]	cfsetispeed(GLIBC_2.0) [1]	tcdrain(GLIBC_2.0) [1]	tcgetattr(GLIBC_2.0) [1]	tcsendbreak(GLIBC_2.0) [1]
cfgetospeed(GLIBC_2.0) [1]	cfsetospeed(GLIBC_2.0) [1]	tcflow(GLIBC_2.0) [1]	tcgetpgrp(GLIBC_2.0) [1]	tcsetattr(GLIBC_2.0) [1]
cfmakeraw(GLIBC_2.0) [2]	cfsetspeed(GLIBC_2.0) [2]	tcflush(GLIBC_2.0) [1]	tcgetsid(GLIBC_2.1) [1]	tcsetpgrp(GLIBC_2.0) [1]

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

1.2.14. System Database Interface

1.2.14.1. Interfaces for System Database Interface

An LSB conforming implementation shall provide the architecture specific functions for System Database Interface specified in Table 1-23, with the full functionality as described in the referenced underlying specification.

Table 1-23. libc - System Database Interface Function Interfaces

endgrent(GLIBC_2.0) [1]	getgrgid(GLIBC_2.0) [1]	getprotobynumber(GLIBC_2.0) [1]	getservbyport(GLIBC_2.0) [1]	setgrent(GLIBC_2.0) [1]
endnetent(GLIBC_2.0) [1]	getgrgid_r(GLIBC_2.1.2) [1]	getprotoent(GLIBC_2.0) [1]	getservent(GLIBC_2.0) [1]	setgroups(GLIBC_2.0) [2]
endprotoent(GLIBC_2.0) [1]	getgrnam(GLIBC_2.0) [1]	getpwent(GLIBC_2.0) [1]	getutent(GLIBC_2.0) [2]	setnetent(GLIBC_2.0) [1]
endpwent(GLIBC_2.0) [1]	getgrnam_r(GLIBC_2.1.2) [1]	getpwnam(GLIBC_2.0) [1]	getutent_r(GLIBC_2.0) [2]	setprotoent(GLIBC_2.0) [1]
endservent(GLIBC_2.0) [1]	gethostbyaddr(GLIBC_2.0) [1]	getpwnam_r(GLIBC_2.1.2) [1]	getutxent(GLIBC_2.1) [1]	setpwent(GLIBC_2.0) [1]
endutent(GLIBC_2.0) [3]	gethostbyname(GLIBC_2.0) [1]	getpwuid(GLIBC_2.0) [1]	getutxid(GLIBC_2.1) [1]	setservent(GLIBC_2.0) [1]
endutxent(GLIBC_2.1) [1]	getnetbyaddr(GLIBC_2.0) [1]	getpwuid_r(GLIBC_2.1.2) [1]	getutxline(GLIBC_2.1) [1]	setutent(GLIBC_2.0) [2]
getgrent(GLIBC_2.0) [1]	getprotobyname(GLIBC_2.0) [1]	getservbyname(GLIBC_2.0) [1]	pututxline(GLIBC_2.1) [1]	setutxent(GLIBC_2.1) [1]

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

[3]. SUSv2

1.2.15. Language Support

1.2.15.1. Interfaces for Language Support

An LSB conforming implementation shall provide the architecture specific functions for Language Support specified in Table 1-24, with the full functionality as described in the referenced underlying specification.

Table 1-24. libc - Language Support Function Interfaces

__libc_start_main(GLIBC_2.0) [1]	_obstack_begin(GLIBC_2.0) [1]	_obstack_newchunk(GLIBC_2.0) [1]	obstack_free(GLIBC_2.0) [1]	

Referenced Specification(s)

[1]. this specification

1.2.16. Large File Support

1.2.16.1. Interfaces for Large File Support

An LSB conforming implementation shall provide the architecture specific functions for Large File Support specified in Table 1-25, with the full functionality as described in the referenced underlying specification.

Table 1-25. libc - Large File Support Function Interfaces

__fxstat64(GLIBC_2.2) [1]	fopen64(GLIBC_2.1) [2]	ftello64(GLIBC_2.1) [2]	lseek64(GLIBC_2.1) [2]	readdir64(GLIBC_2.2) [2]
__lxstat64(GLIBC_2.2) [1]	freopen64(GLIBC_2.1) [2]	ftruncate64(GLIBC_2.1) [2]	mkstemp64(GLIBC_2.2) [2]	statvfs64(GLIBC_2.1) [2]
__xstat64(GLIBC_2.2) [1]	fseeko64(GLIBC_2.1) [2]	ftw64(GLIBC_2.1) [2]	mmap64(GLIBC_2.1) [2]	tmpfile64(GLIBC_2.1) [2]
creat64(GLIBC_2.1) [2]	fsetpos64(GLIBC_2.2) [2]	getrlimit64(GLIBC_2.2) [2]	nftw64(GLIBC_2.1) [2]	truncate64(GLIBC_2.1) [2]
fgetpos64(GLIBC_2.2) [2]	fstatvfs64(GLIBC_2.1) [2]	lockf64(GLIBC_2.1) [2]	open64(GLIBC_2.1) [2]	

Referenced Specification(s)

[1]. this specification

[2]. Large File Support

1.2.17. Standard Library

1.2.17.1. Interfaces for Standard Library

An LSB conforming implementation shall provide the architecture specific functions for Standard Library specified in Table 1-26, with the full functionality as described in the referenced underlying specification.

Table 1-26. libc - Standard Library Function Interfaces

_Exit(GLIBC_2.1.1) [1]	dirname(GLIBC_2.0) [1]	glob(GLIBC_2.0) [1]	lsearch(GLIBC_2.0) [1]	srand(GLIBC_2.0) [1]
__assert_fail(GLIBC_2.0) [2]	div(GLIBC_2.0) [1]	glob64(GLIBC_2.2) [2]	makecontext(GLIBC_2.1) [1]	srand48(GLIBC_2.0) [1]
__cxa_atexit(GLIBC_2.1.3) [2]	drand48(GLIBC_2.0) [1]	globfree(GLIBC_2.0) [1]	malloc(GLIBC_2.0) [1]	srandom(GLIBC_2.0) [1]
__errno_location(GLIBC_2.0) [2]	ecvt(GLIBC_2.0) [1]	globfree64(GLIBC_2.1) [2]	memmem(GLIBC_2.0) [2]	strtod(GLIBC_2.0) [1]

__fpending(GLIBC_2.2) [2]	erand48(GLIBC_2.0) [1]	grantpt(GLIBC_2.1) [1]	mkstemp(GLIBC_2.0) [1]	strtol(GLIBC_2.0) [1]
__getpagesize(GLIBC_2.0) [2]	err(GLIBC_2.0) [2]	hcreate(GLIBC_2.0) [1]	mktemp(GLIBC_2.0) [1]	strtoul(GLIBC_2.0) [1]
__isinf(GLIBC_2.0) [2]	error(GLIBC_2.0) [2]	hdestroy(GLIBC_2.0) [1]	mrand48(GLIBC_2.0) [1]	swapcontext(GLIBC_2.1) [1]
__isinff(GLIBC_2.0) [2]	errx(GLIBC_2.0) [2]	hsearch(GLIBC_2.0) [1]	nftw(GLIBC_2.1) [1]	syslog(GLIBC_2.0) [1]
__isinfl(GLIBC_2.0) [2]	fcvt(GLIBC_2.0) [1]	htonl(GLIBC_2.0) [1]	nrand48(GLIBC_2.0) [1]	system(GLIBC_2.0) [2]
__isnan(GLIBC_2.0) [2]	fmtmsg(GLIBC_2.1) [1]	htons(GLIBC_2.0) [1]	ntohl(GLIBC_2.0) [1]	tdelete(GLIBC_2.0) [1]
__isnanf(GLIBC_2.0) [2]	fnmatch(GLIBC_2.2.3) [1]	imaxabs(GLIBC_2.1.1) [1]	ntohs(GLIBC_2.0) [1]	tfind(GLIBC_2.0) [1]
__isnanl(GLIBC_2.0) [2]	fpathconf(GLIBC_2.0) [1]	imaxdiv(GLIBC_2.1.1) [1]	openlog(GLIBC_2.0) [1]	tmpfile(GLIBC_2.1) [1]
__sysconf(GLIBC_2.2) [2]	free(GLIBC_2.0) [1]	inet_addr(GLIBC_2.0) [1]	perror(GLIBC_2.0) [1]	tmpnam(GLIBC_2.0) [1]
_exit(GLIBC_2.0) [1]	freeaddrinfo(GLIBC_2.0) [1]	inet_ntoa(GLIBC_2.0) [1]	posix_memalign(GLIBC_2.2) [1]	tsearch(GLIBC_2.0) [1]
_longjmp(GLIBC_2.0) [1]	ftrylockfile(GLIBC_2.0) [1]	inet_ntop(GLIBC_2.0) [1]	ptsname(GLIBC_2.1) [1]	ttyname(GLIBC_2.0) [1]
_setjmp(GLIBC_2.0) [1]	ftw(GLIBC_2.0) [1]	inet_pton(GLIBC_2.0) [1]	putenv(GLIBC_2.0) [1]	ttyname_r(GLIBC_2.0) [1]
a64l(GLIBC_2.0) [1]	funlockfile(GLIBC_2.0) [1]	initstate(GLIBC_2.0) [1]	qsort(GLIBC_2.0) [1]	twalk(GLIBC_2.0) [1]
abort(GLIBC_2.0) [1]	gai_strerror(GLIBC_2.1) [1]	insque(GLIBC_2.0) [1]	rand(GLIBC_2.0) [1]	unlockpt(GLIBC_2.1) [1]
abs(GLIBC_2.0) [1]	gcvt(GLIBC_2.0) [1]	isatty(GLIBC_2.0) [1]	rand_r(GLIBC_2.0) [1]	unsetenv(GLIBC_2.0) [1]
atof(GLIBC_2.0) [1]	getaddrinfo(GLIBC_2.0) [1]	isblank(GLIBC_2.0) [1]	random(GLIBC_2.0) [1]	usleep(GLIBC_2.0) [1]
atoi(GLIBC_2.0) [1]	getcwd(GLIBC_2.0) [1]	jrand48(GLIBC_2.0) [1]	random_r(GLIBC_2.0) [2]	verrx(GLIBC_2.0) [2]
atol(GLIBC_2.0) [1]	getdate(GLIBC_2.1) [1]	l64a(GLIBC_2.0) [1]	realloc(GLIBC_2.0) [1]	vfscanf(GLIBC_2.0) [1]
atoll(GLIBC_2.0)	getenv(GLIBC_2.0)	labs(GLIBC_2.0)	realpath(GLIBC_2.	vscanf(GLIBC_2.0)

[1]	[1]	[1]	3) [1]	[1]
basename(GLIBC_2.0) [1]	getlogin(GLIBC_2.0) [1]	lcong48(GLIBC_2.0) [1]	remque(GLIBC_2.0) [1]	vsscanf(GLIBC_2.0) [1]
bsearch(GLIBC_2.0) [1]	getnameinfo(GLIBC_2.1) [1]	ldiv(GLIBC_2.0) [1]	seed48(GLIBC_2.0) [1]	vsyslog(GLIBC_2.0) [2]
calloc(GLIBC_2.0) [1]	getopt(GLIBC_2.0) [2]	lfind(GLIBC_2.0) [1]	setenv(GLIBC_2.0) [1]	warn(GLIBC_2.0) [2]
closelog(GLIBC_2.0) [1]	getopt_long(GLIBC_2.0) [2]	llabs(GLIBC_2.0) [1]	sethostid(GLIBC_2.0) [2]	warnx(GLIBC_2.0) [2]
confstr(GLIBC_2.0) [1]	getopt_long_only(GLIBC_2.0) [2]	lldiv(GLIBC_2.0) [1]	sethostname(GLIBC_2.0) [2]	wordexp(GLIBC_2.1) [1]
cuserid(GLIBC_2.0) [3]	getsubopt(GLIBC_2.0) [1]	longjmp(GLIBC_2.0) [1]	setlogmask(GLIBC_2.0) [1]	wordfree(GLIBC_2.1) [1]
daemon(GLIBC_2.0) [2]	gettimeofday(GLIBC_2.0) [1]	lrand48(GLIBC_2.0) [1]	setstate(GLIBC_2.0) [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

[3]. SUSv2

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard Library specified in Table 1-27, with the full functionality as described in the referenced underlying specification.

Table 1-27. libc - Standard Library Data Interfaces

__environ(GLIBC_2.0) [1]	_sys_errlist(GLIBC_2.3) [1]	getdate_err(GLIBC_2.1) [2]	opterr(GLIBC_2.0) [1]	optopt(GLIBC_2.0) [1]
_environ(GLIBC_2.0) [1]	environ(GLIBC_2.0) [2]	optarg(GLIBC_2.0) [2]	optind(GLIBC_2.0) [1]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

1.3. 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 the referenced underlying specifications.

This specification uses ISO/IEC 9899 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.

1.3.1. errno.h

```

#define EDEADLOCK       EDEADLK
```

1.3.2. inttypes.h

```

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

1.3.3. limits.h

```

#define LONG_MAX        0x7FFFFFFFL
#define ULONG_MAX       0xFFFFFFFFUL

#define CHAR_MAX        SCHAR_MAX
#define CHAR_MIN        SCHAR_MIN
```

1.3.4. setjmp.h

```

typedef int __jmp_buf[6];
```

1.3.5. signal.h

```

struct sigaction
{
  union
  {
    sighandler_t _sa_handler;
    void (*_sa_sigaction) (int, siginfo_t *, void *);
  }
  __sigaction_handler;
  sigset_t sa_mask;
  unsigned long sa_flags;
  void (*sa_restorer) (void);
}
 ;
#define MINSIGSTKSZ     2048
#define SIGSTKSZ        8192

struct _fpreg
```

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

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

struct sigcontext
{
  unsigned short gs;
  unsigned short __gsh;
  unsigned short fs;
  unsigned short __fsh;
  unsigned short es;
  unsigned short __esh;
  unsigned short ds;
  unsigned short __dsh;
  unsigned long edi;
  unsigned long esi;
  unsigned long ebp;
  unsigned long esp;
  unsigned long ebx;
  unsigned long edx;
  unsigned long ecx;
  unsigned long eax;
  unsigned long trapno;
  unsigned long err;
  unsigned long eip;
  unsigned short cs;
  unsigned short __csh;
  unsigned long eflags;
  unsigned long esp_at_signal;
  unsigned short ss;
  unsigned short __ssh;
  struct _fpstate *fpstate;
  unsigned long oldmask;
  unsigned long cr2;
}
 ;
```

1.3.6. stddef.h

```

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

1.3.7. sys/ioctl.h

```

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

1.3.8. sys/ipc.h

```

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

1.3.9. sys/mman.h

```

#define MCL_CURRENT     1
#define MCL_FUTURE      2
```

1.3.10. sys/msg.h

```

typedef unsigned long msgqnum_t;
typedef unsigned long msglen_t;

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

1.3.11. sys/sem.h

```

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

1.3.12. sys/shm.h

```

#define SHMLBA  (__getpagesize())
```

```

typedef unsigned long shmatt_t;

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

1.3.13. sys/socket.h

```

typedef uint32_t __ss_aligntype;
```

1.3.14. sys/stat.h

```

#define _STAT_VER       3

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

```
struct stat64
{
  dev_t st_dev;
  unsigned int __pad1;
  ino_t __st_ino;
  mode_t st_mode;
  nlink_t st_nlink;
  uid_t st_uid;
  gid_t st_gid;
  dev_t st_rdev;
  unsigned int __pad2;
  off64_t st_size;
  blksize_t st_blksize;
  blkcnt64_t st_blocks;
  struct timespec st_atim;
  struct timespec st_mtim;
  struct timespec st_ctim;
  ino64_t st_ino;
}
 ;
```

1.3.15. sys/statvfs.h

```

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

1.3.16. sys/types.h

```

typedef long long int64_t;

typedef int32_t ssize_t;
```

1.3.17. 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
```

1.3.18. ucontext.h

```

typedef int greg_t;
#define NGREG   19

typedef greg_t gregset_t[19];

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

struct _libc_fpstate
{
  unsigned long cw;
  unsigned long sw;
  unsigned long tag;
  unsigned long ipoff;
  unsigned long cssel;
  unsigned long dataoff;
  unsigned long datasel;
  struct _libc_fpreg _st[8];
  unsigned long status;
}
```

```
 ;
typedef struct _libc_fpstate *fpregset_t;

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

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

1.3.19. unistd.h

```

typedef int intptr_t;
```

1.3.20. 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 ut_session;
  struct timeval ut_tv;
  int32_t ut_addr_v6[4];
  char __unused[20];
}
 ;
```

1.3.21. 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 ut_session;
  struct timeval ut_tv;
  int32_t ut_addr_v6[4];
  char __unused[20];
}
 ;
```

1.4. Interfaces for libm

Table 1-28 defines the library name and shared object name for the libm library

Table 1-28. libm Definition

Library:	libm
SONAME:	libm.so.6

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

ISO C (1999)
SUSv2
ISO POSIX (2003)

1.4.1. Math

1.4.1.1. Interfaces for Math

An LSB conforming implementation shall provide the architecture specific functions for Math specified in Table 1-29, with the full functionality as described in the referenced underlying specification.

Table 1-29. libm - Math Function Interfaces

acos(GLIBC_2.0) [1]	cexp(GLIBC_2.1) [1]	expf(GLIBC_2.0) [1]	jnf(GLIBC_2.0) [2]	remquof(GLIBC_2.1) [1]
acosf(GLIBC_2.0)	cexpf(GLIBC_2.1)	expl(GLIBC_2.0)	jnl(GLIBC_2.0) [2]	remquol(GLIBC_2.

[1]	[1]	[1]		1) [1]
acosh(GLIBC_2.0) [1]	cexpl(GLIBC_2.1) [1]	expm1(GLIBC_2.0) [1]	ldexp(GLIBC_2.0) [1]	rint(GLIBC_2.0) [1]
acoshf(GLIBC_2.0) [1]	cimag(GLIBC_2.1) [1]	fabs(GLIBC_2.0) [1]	ldexpf(GLIBC_2.0) [1]	rintf(GLIBC_2.0) [1]
acoshl(GLIBC_2.0) [1]	cimagf(GLIBC_2.1) [1]	fabsf(GLIBC_2.0) [1]	ldexpl(GLIBC_2.0) [1]	rintl(GLIBC_2.0) [1]
acosl(GLIBC_2.0) [1]	cimagl(GLIBC_2.1) [1]	fabsl(GLIBC_2.0) [1]	lgamma(GLIBC_2.0) [1]	round(GLIBC_2.1) [1]
asin(GLIBC_2.0) [1]	clog(GLIBC_2.1) [1]	fdim(GLIBC_2.1) [1]	lgamma_r(GLIBC_2.0) [2]	roundf(GLIBC_2.1) [1]
asinf(GLIBC_2.0) [1]	clog10(GLIBC_2.1) [2]	fdimf(GLIBC_2.1) [1]	lgammaf(GLIBC_2.0) [1]	roundl(GLIBC_2.1) [1]
asinh(GLIBC_2.0) [1]	clog10f(GLIBC_2.1) [2]	fdiml(GLIBC_2.1) [1]	lgammaf_r(GLIBC_2.0) [2]	scalb(GLIBC_2.0) [1]
asinhf(GLIBC_2.0) [1]	clog10l(GLIBC_2.1) [2]	feclearexcept(GLIBC_2.2) [1]	lgammal(GLIBC_2.0) [1]	scalbf(GLIBC_2.0) [2]
asinhl(GLIBC_2.0) [1]	clogf(GLIBC_2.1) [1]	fegetenv(GLIBC_2.2) [1]	lgammal_r(GLIBC_2.0) [2]	scalbl(GLIBC_2.0) [2]
asinl(GLIBC_2.0) [1]	clogl(GLIBC_2.1) [1]	fegetexceptflag(GLIBC_2.2) [1]	llrint(GLIBC_2.1) [1]	scalbln(GLIBC_2.1) [1]
atan(GLIBC_2.0) [1]	conj(GLIBC_2.1) [1]	fegetround(GLIBC_2.1) [1]	llrintf(GLIBC_2.1) [1]	scalblnf(GLIBC_2.1) [1]
atan2(GLIBC_2.0) [1]	conjf(GLIBC_2.1) [1]	feholdexcept(GLIBC_2.1) [1]	llrintl(GLIBC_2.1) [1]	scalblnl(GLIBC_2.1) [1]
atan2f(GLIBC_2.0) [1]	conjl(GLIBC_2.1) [1]	feraiseexcept(GLIBC_2.2) [1]	llround(GLIBC_2.1) [1]	scalbn(GLIBC_2.0) [1]
atan2l(GLIBC_2.0) [1]	copysign(GLIBC_2.0) [1]	fesetenv(GLIBC_2.2) [1]	llroundf(GLIBC_2.1) [1]	scalbnf(GLIBC_2.0) [1]
atanf(GLIBC_2.0) [1]	copysignf(GLIBC_2.0) [1]	fesetexceptflag(GLIBC_2.2) [1]	llroundl(GLIBC_2.1) [1]	scalbnl(GLIBC_2.0) [1]
atanh(GLIBC_2.0) [1]	copysignl(GLIBC_2.0) [1]	fesetround(GLIBC_2.1) [1]	log(GLIBC_2.0) [1]	significand(GLIBC_2.0) [2]
atanhf(GLIBC_2.0) [1]	cos(GLIBC_2.0) [1]	fetestexcept(GLIBC_2.1) [1]	log10(GLIBC_2.0) [1]	significandf(GLIBC_2.0) [2]
atanhl(GLIBC_2.0) [1]	cosf(GLIBC_2.0) [1]	feupdateenv(GLIBC_2.2) [1]	log10f(GLIBC_2.0) [1]	significandl(GLIBC_2.0) [2]

atanl(GLIBC_2.0) [1]	cosh(GLIBC_2.0) [1]	finite(GLIBC_2.0) [3]	log10l(GLIBC_2.0) [1]	sin(GLIBC_2.0) [1]
cabs(GLIBC_2.1) [1]	coshf(GLIBC_2.0) [1]	finitef(GLIBC_2.0) [2]	log1p(GLIBC_2.0) [1]	sincos(GLIBC_2.1) [2]
cabsf(GLIBC_2.1) [1]	coshl(GLIBC_2.0) [1]	finitel(GLIBC_2.0) [2]	logb(GLIBC_2.0) [1]	sincosf(GLIBC_2.1) [2]
cabsl(GLIBC_2.1) [1]	cosl(GLIBC_2.0) [1]	floor(GLIBC_2.0) [1]	logf(GLIBC_2.0) [1]	sincosl(GLIBC_2.1) [2]
cacos(GLIBC_2.1) [1]	cpow(GLIBC_2.1) [1]	floorf(GLIBC_2.0) [1]	logl(GLIBC_2.0) [1]	sinf(GLIBC_2.0) [1]
cacosf(GLIBC_2.1) [1]	cpowf(GLIBC_2.1) [1]	floorl(GLIBC_2.0) [1]	lrint(GLIBC_2.1) [1]	sinh(GLIBC_2.0) [1]
cacosh(GLIBC_2.1) [1]	cpowl(GLIBC_2.1) [1]	fma(GLIBC_2.1) [1]	lrintf(GLIBC_2.1) [1]	sinhf(GLIBC_2.0) [1]
cacoshf(GLIBC_2.1) [1]	cproj(GLIBC_2.1) [1]	fmaf(GLIBC_2.1) [1]	lrintl(GLIBC_2.1) [1]	sinhl(GLIBC_2.0) [1]
cacoshl(GLIBC_2.1) [1]	cprojf(GLIBC_2.1) [1]	fmal(GLIBC_2.1) [1]	lround(GLIBC_2.1) [1]	sinl(GLIBC_2.0) [1]
cacosl(GLIBC_2.1) [1]	cprojl(GLIBC_2.1) [1]	fmax(GLIBC_2.1) [1]	lroundf(GLIBC_2.1) [1]	sqrt(GLIBC_2.0) [1]
carg(GLIBC_2.1) [1]	creal(GLIBC_2.1) [1]	fmaxf(GLIBC_2.1) [1]	lroundl(GLIBC_2.1) [1]	sqrtf(GLIBC_2.0) [1]
cargf(GLIBC_2.1) [1]	crealf(GLIBC_2.1) [1]	fmaxl(GLIBC_2.1) [1]	matherr(GLIBC_2.0) [2]	sqrtl(GLIBC_2.0) [1]
cargl(GLIBC_2.1) [1]	creall(GLIBC_2.1) [1]	fmin(GLIBC_2.1) [1]	modf(GLIBC_2.0) [1]	tan(GLIBC_2.0) [1]
casin(GLIBC_2.1) [1]	csin(GLIBC_2.1) [1]	fminf(GLIBC_2.1) [1]	modff(GLIBC_2.0) [1]	tanf(GLIBC_2.0) [1]
casinf(GLIBC_2.1) [1]	csinf(GLIBC_2.1) [1]	fminl(GLIBC_2.1) [1]	modfl(GLIBC_2.0) [1]	tanh(GLIBC_2.0) [1]
casinh(GLIBC_2.1) [1]	csinh(GLIBC_2.1) [1]	fmod(GLIBC_2.0) [1]	nan(GLIBC_2.1) [1]	tanhf(GLIBC_2.0) [1]
casinhf(GLIBC_2.1) [1]	csinhf(GLIBC_2.1) [1]	fmodf(GLIBC_2.0) [1]	nanf(GLIBC_2.1) [1]	tanhl(GLIBC_2.0) [1]
casinhl(GLIBC_2.1) [1]	csinhl(GLIBC_2.1) [1]	fmodl(GLIBC_2.0) [1]	nanl(GLIBC_2.1) [1]	tanl(GLIBC_2.0) [1]
casinl(GLIBC_2.1)	csinl(GLIBC_2.1)	frexp(GLIBC_2.0)	nearbyint(GLIBC_2	tgamma(GLIBC_2.

[1]	[1]	[1]	.1) [1]	1) [1]
catan(GLIBC_2.1) [1]	csqrt(GLIBC_2.1) [1]	frexpf(GLIBC_2.0) [1]	nearbyintf(GLIBC_2.1) [1]	tgammaf(GLIBC_2.1) [1]
catanf(GLIBC_2.1) [1]	csqrtf(GLIBC_2.1) [1]	frexpl(GLIBC_2.0) [1]	nearbyintl(GLIBC_2.1) [1]	tgammal(GLIBC_2.1) [1]
catanh(GLIBC_2.1) [1]	csqrtl(GLIBC_2.1) [1]	gamma(GLIBC_2.0) [3]	nextafter(GLIBC_2.0) [1]	trunc(GLIBC_2.1) [1]
catanhf(GLIBC_2.1) [1]	ctan(GLIBC_2.1) [1]	gammaf(GLIBC_2.0) [2]	nextafterf(GLIBC_2.0) [1]	truncf(GLIBC_2.1) [1]
catanhl(GLIBC_2.1) [1]	ctanf(GLIBC_2.1) [1]	gammal(GLIBC_2.0) [2]	nextafterl(GLIBC_2.0) [1]	truncl(GLIBC_2.1) [1]
catanl(GLIBC_2.1) [1]	ctanh(GLIBC_2.1) [1]	hypot(GLIBC_2.0) [1]	nexttoward(GLIBC_2.1) [1]	y0(GLIBC_2.0) [1]
cbrt(GLIBC_2.0) [1]	ctanhf(GLIBC_2.1) [1]	hypotf(GLIBC_2.0) [1]	nexttowardf(GLIBC_2.1) [1]	y0f(GLIBC_2.0) [2]
cbrtf(GLIBC_2.0) [1]	ctanhl(GLIBC_2.1) [1]	hypotl(GLIBC_2.0) [1]	nexttowardl(GLIBC_2.1) [1]	y0l(GLIBC_2.0) [2]
cbrtl(GLIBC_2.0) [1]	ctanl(GLIBC_2.1) [1]	ilogb(GLIBC_2.0) [1]	pow(GLIBC_2.0) [1]	y1(GLIBC_2.0) [1]
ccos(GLIBC_2.1) [1]	dremf(GLIBC_2.0) [2]	ilogbf(GLIBC_2.0) [1]	pow10(GLIBC_2.1) [2]	y1f(GLIBC_2.0) [2]
ccosf(GLIBC_2.1) [1]	dreml(GLIBC_2.0) [2]	ilogbl(GLIBC_2.0) [1]	pow10f(GLIBC_2.1) [2]	y1l(GLIBC_2.0) [2]
ccosh(GLIBC_2.1) [1]	erf(GLIBC_2.0) [1]	j0(GLIBC_2.0) [1]	pow10l(GLIBC_2.1) [2]	yn(GLIBC_2.0) [1]
ccoshf(GLIBC_2.1) [1]	erfc(GLIBC_2.0) [1]	j0f(GLIBC_2.0) [2]	powf(GLIBC_2.0) [1]	ynf(GLIBC_2.0) [2]
ccoshl(GLIBC_2.1) [1]	erfcf(GLIBC_2.0) [1]	j0l(GLIBC_2.0) [2]	powl(GLIBC_2.0) [1]	ynl(GLIBC_2.0) [2]
ccosl(GLIBC_2.1) [1]	erfcl(GLIBC_2.0) [1]	j1(GLIBC_2.0) [1]	remainder(GLIBC_2.0) [1]	
ceil(GLIBC_2.0) [1]	erff(GLIBC_2.0) [1]	j1f(GLIBC_2.0) [2]	remainderf(GLIBC_2.0) [1]	
ceilf(GLIBC_2.0) [1]	erfl(GLIBC_2.0) [1]	j1l(GLIBC_2.0) [2]	remainderl(GLIBC_2.0) [1]	
ceill(GLIBC_2.0) [1]	exp(GLIBC_2.0) [1]	jn(GLIBC_2.0) [1]	remquo(GLIBC_2.1) [1]	

652

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. ISO C (1999)

[3]. SUSv2

An LSB conforming implementation shall provide the architecture specific data interfaces for Math specified in Table 1-30, with the full functionality as described in the referenced underlying specification.

Table 1-30. libm - Math Data Interfaces

signgam(GLIBC_2.0) [1]				

Referenced Specification(s)

[1]. ISO POSIX (2003)

1.5. Interfaces for libpthread

Table 1-31 defines the library name and shared object name for the libpthread library

Table 1-31. libpthread Definition

Library:	libpthread
SONAME:	libpthread.so.0

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

Large File Support
this specification
ISO POSIX (2003)

1.5.1. Realtime Threads

1.5.1.1. Interfaces for Realtime Threads

No external functions are defined for libpthread - Realtime Threads

1.5.2. Advanced Realtime Threads

1.5.2.1. Interfaces for Advanced Realtime Threads

No external functions are defined for libpthread - Advanced Realtime Threads

1.5.3. Posix Threads

1.5.3.1. Interfaces for Posix Threads

An LSB conforming implementation shall provide the architecture specific functions for Posix Threads specified in Table 1-32, with the full functionality as described in the referenced underlying specification.

Table 1-32. libpthread - Posix Threads Function Interfaces

_pthread_cleanup_pop(GLIBC_2.0) [1]	pthread_cancel(GLIBC_2.0) [2]	pthread_join(GLIBC_2.0) [2]	pthread_rwlock_destroy(GLIBC_2.1) [2]	pthread_setconcurrency(GLIBC_2.1) [2]
_pthread_cleanup_push(GLIBC_2.0) [1]	pthread_cond_broadcast(GLIBC_2.3.2) [2]	pthread_key_create(GLIBC_2.0) [2]	pthread_rwlock_init(GLIBC_2.1) [2]	pthread_setspecific(GLIBC_2.0) [2]
pread(GLIBC_2.2) [2]	pthread_cond_destroy(GLIBC_2.3.2) [2]	pthread_key_delete(GLIBC_2.0) [2]	pthread_rwlock_rdlock(GLIBC_2.1) [2]	pthread_sigmask(GLIBC_2.0) [2]
pread64(GLIBC_2.2) [3]	pthread_cond_init(GLIBC_2.3.2) [2]	pthread_kill(GLIBC_2.0) [2]	pthread_rwlock_timedrdlock(GLIBC_2.2) [2]	pthread_testcancel(GLIBC_2.0) [2]
pthread_attr_destroy(GLIBC_2.0) [2]	pthread_cond_signal(GLIBC_2.3.2) [2]	pthread_mutex_destroy(GLIBC_2.0) [2]	pthread_rwlock_timedwrlock(GLIBC_2.2) [2]	pwrite(GLIBC_2.2) [2]
pthread_attr_getdetachstate(GLIBC_2.0) [2]	pthread_cond_timedwait(GLIBC_2.3.2) [2]	pthread_mutex_init(GLIBC_2.0) [2]	pthread_rwlock_tryrdlock(GLIBC_2.1) [2]	pwrite64(GLIBC_2.2) [3]
pthread_attr_getguardsize(GLIBC_2.1) [2]	pthread_cond_wait(GLIBC_2.3.2) [2]	pthread_mutex_lock(GLIBC_2.0) [2]	pthread_rwlock_trywrlock(GLIBC_2.1) [2]	sem_close(GLIBC_2.1.1) [2]
pthread_attr_getschedparam(GLIBC_2.0) [2]	pthread_condattr_destroy(GLIBC_2.0) [2]	pthread_mutex_trylock(GLIBC_2.0) [2]	pthread_rwlock_unlock(GLIBC_2.1) [2]	sem_destroy(GLIBC_2.1) [2]
pthread_attr_getstackaddr(GLIBC_2.1) [2]	pthread_condattr_getpshared(GLIBC_2.2) [2]	pthread_mutex_unlock(GLIBC_2.0) [2]	pthread_rwlock_wrlock(GLIBC_2.1) [2]	sem_getvalue(GLIBC_2.1) [2]
pthread_attr_getstacksize(GLIBC_2.1) [2]	pthread_condattr_init(GLIBC_2.0) [2]	pthread_mutexattr_destroy(GLIBC_2.0) [2]	pthread_rwlockattr_destroy(GLIBC_2.1) [2]	sem_init(GLIBC_2.1) [2]
pthread_attr_init(GLIBC_2.1) [2]	pthread_condattr_setpshared(GLIBC_2.2) [2]	pthread_mutexattr_getpshared(GLIBC_2.2) [2]	pthread_rwlockattr_getpshared(GLIBC_2.1) [2]	sem_open(GLIBC_2.1.1) [2]

pthread_attr_setdetachstate(GLIBC_2.0) [2]	pthread_create(GLIBC_2.1) [2]	pthread_mutexattr_gettype(GLIBC_2.1) [2]	pthread_rwlockattr_init(GLIBC_2.1) [2]	sem_post(GLIBC_2.1) [2]
pthread_attr_setguardsize(GLIBC_2.1) [2]	pthread_detach(GLIBC_2.0) [2]	pthread_mutexattr_init(GLIBC_2.0) [2]	pthread_rwlockattr_setpshared(GLIBC_2.1) [2]	sem_timedwait(GLIBC_2.2) [2]
pthread_attr_setschedparam(GLIBC_2.0) [2]	pthread_equal(GLIBC_2.0) [2]	pthread_mutexattr_setpshared(GLIBC_2.2) [2]	pthread_self(GLIBC_2.0) [2]	sem_trywait(GLIBC_2.1) [2]
pthread_attr_setstackaddr(GLIBC_2.1) [2]	pthread_exit(GLIBC_2.0) [2]	pthread_mutexattr_settype(GLIBC_2.1) [2]	pthread_setcancelstate(GLIBC_2.0) [2]	sem_unlink(GLIBC_2.1.1) [2]
pthread_attr_setstacksize(GLIBC_2.1) [2]	pthread_getspecific(GLIBC_2.0) [2]	pthread_once(GLIBC_2.0) [2]	pthread_setcanceltype(GLIBC_2.0) [2]	sem_wait(GLIBC_2.1) [2]

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. Large File Support

1.6. Interfaces for libgcc_s

Table 1-33 defines the library name and shared object name for the libgcc_s library

Table 1-33. 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:

this specification

1.6.1. Unwind Library

1.6.1.1. Interfaces for Unwind Library

An LSB conforming implementation shall provide the architecture specific functions for Unwind Library specified in Table 1-34, with the full functionality as described in the referenced underlying specification.

Table 1-34. libgcc_s - Unwind Library Function Interfaces

_Unwind_DeleteException(GCC_3.0)	_Unwind_GetDataRelBase(GCC_3.0)	_Unwind_GetLanguageSpecificData(G	_Unwind_RaiseException(GCC_3.0)	_Unwind_SetIP(GC

[1]	[1]	CC_3.0) [1]	[1]	C_3.0) [1]
_Unwind_Find_FDE(GCC_3.0) [1]	_Unwind_GetGR(GCC_3.0) [1]	_Unwind_GetRegionStart(GCC_3.0) [1]	_Unwind_Resume(GCC_3.0) [1]	
_Unwind_ForcedUnwind(GCC_3.0) [1]	_Unwind_GetIP(GCC_3.0) [1]	_Unwind_GetTextRelBase(GCC_3.0) [1]	_Unwind_SetGR(GCC_3.0) [1]	

Referenced Specification(s)

[1]. this specification

1.7. 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.

_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_Find_FDE

Name

702 `_Unwind_Find_FDE` — private C++ error handling method

Synopsis

703
```
fde * _Unwind_Find_FDE(void *pc, (struct dwarf_eh_bases *bases));
```

Description

704 `_Unwind_Find_FDE` looks for the object containing *`pc`*, then inserts into *`bases`*.

_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_GetDataRelBase

Name

`_Unwind_GetDataRelBase` — private IA64 C++ error handling method

Synopsis

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

Description

`_Unwind_GetDataRelBase` returns the global pointer in register one for *`context`*.

_Unwind_GetGR

Name

`_Unwind_GetGR` — private C++ error handling method

Synopsis

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

Description

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

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

_Unwind_GetIP

Name

`_Unwind_GetIP` — private C++ error handling method

Synopsis

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

Description

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

_Unwind_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_GetTextRelBase

Name

`_Unwind_GetTextRelBase` — private IA64 C++ error handling method

Synopsis

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

Description

`_Unwind_GetTextRelBase` calls the abort method, then returns.

_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`*

1.8. Interfaces for libdl

Table 1-35 defines the library name and shared object name for the libdl library

Table 1-35. libdl Definition

Library:	libdl
SONAME:	libdl.so.2

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

this specification
ISO POSIX (2003)

1.8.1. Dynamic Loader

1.8.1.1. Interfaces for Dynamic Loader

An LSB conforming implementation shall provide the architecture specific functions for Dynamic Loader specified in Table 1-36, with the full functionality as described in the referenced underlying specification.

Table 1-36. 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]. this specification

[2]. ISO POSIX (2003)

1.9. Interfaces for libcrypt

Table 1-37 defines the library name and shared object name for the libcrypt library

Table 1-37. libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

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

ISO POSIX (2003)

1.9.1. Encryption

1.9.1.1. Interfaces for Encryption

An LSB conforming implementation shall provide the architecture specific functions for Encryption specified in Table 1-38, with the full functionality as described in the referenced underlying specification.

Table 1-38. libcrypt - Encryption Function Interfaces

crypt(GLIBC_2.0)	encrypt(GLIBC_2.0	setkey(GLIBC_2.0)		

[1]	) [1]	[1]		

Referenced Specification(s)

[1]. ISO POSIX (2003)

II. Utility Libraries

Chapter 2. 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.

2.1. Interfaces for libz

Table 2-1 defines the library name and shared object name for the libz library

Table 2-1. libz Definition

Library:	libz
SONAME:	libz.so.1

2.1.1. Compression Library

2.1.1.1. Interfaces for Compression Library

No external functions are defined for libz - Compression Library

2.2. Interfaces for libncurses

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

Table 2-2. libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

2.2.1. Curses

2.2.1.1. Interfaces for Curses

No external functions are defined for libncurses - Curses

2.3. Interfaces for libutil

Table 2-3 defines the library name and shared object name for the libutil library

Table 2-3. libutil Definition

Library:	libutil
SONAME:	libutil.so.1

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

this specification

2.3.1. Utility Functions

2.3.1.1. Interfaces for Utility Functions

An LSB conforming implementation shall provide the architecture specific functions for Utility Functions specified in Table 2-4, with the full functionality as described in the referenced underlying specification.

Table 2-4. 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

Appendix A. Alphabetical Listing of Interfaces

A.1. libgcc_s

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

this specification

Table A-1. libgcc_s Function Interfaces

_Unwind_DeleteException[1]	_Unwind_GetIP[1]	_Unwind_Resume[1]
_Unwind_Find_FDE[1]	_Unwind_GetLanguageSpecificData[1]	_Unwind_SetGR[1]
_Unwind_ForcedUnwind[1]	_Unwind_GetRegionStart[1]	_Unwind_SetIP[1]
_Unwind_GetDataRelBase[1]	_Unwind_GetTextRelBase[1]	
_Unwind_GetGR[1]	_Unwind_RaiseException[1]	

Linux Packaging Specification

Linux Packaging Specification

Table of Contents

I. Package Format and Installation

Chapter 1. Software Installation

1.1. Package Dependencies

The LSB runtime environment shall provide the following dependencies.

lsb-core-ia32

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

Other LSB modules may add additional dependencies; such dependencies shall have the format `lsb-`*`module`*`-ia32`.

1.2. Package Architecture Considerations

All packages must specify an architecture of `i486`. A LSB runtime environment must accept an architecture of `i486` even if the native architecture is different.

The `archnum` value in the Lead Section shall be 0x0001.

Free Documentation License

Free Documentation License

Table of Contents

Appendix A. GNU Free Documentation License

Version 1.1, March 2000

A.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.

A.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

A.3. VERBATIM COPYING

A.4. COPYING IN QUANTITY

A.5. MODIFICATIONS

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.

A.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."

A.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.

A.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.

A.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.

A.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.

A.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.

A.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.