

# **Linux Standard Base Core Specification for IA32 2.0.1**

## **Linux Standard Base Core Specification for IA32 2.0.1**

Copyright © 2004 Free Standards Group

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Portions of the text are copyrighted by the following parties:

- The Regents of the University of California
- Free Software Foundation
- Ian F. Darwin
- Paul Vixie
- BSDI (now Wind River)
- Andrew G Morgan
- Jean-loup Gailly and Mark Adler
- Massachusetts Institute of Technology

These excerpts are being used in accordance with their respective licenses.

Linux is a trademark of Linus Torvalds.

UNIX a registered trademark of the Open Group in the United States and other countries.

LSB is a trademark of the Free Standards Group in the USA and other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

Intel and Itanium are registered trademarks and Intel386 is a trademarks of Intel Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

# **Specification Introduction**

## **Specification Introduction**

# Table of Contents

<b>Foreword .....</b>	<b>i</b>
<b>Introduction .....</b>	<b>ii</b>
<b>I. Introductory Elements .....</b>	<b>3</b>
1. Scope .....	1
1.1. General .....	1
1.2. Module Specific Scope .....	1
2. Normative References .....	2
3. Requirements .....	5
3.1. Relevant Libraries .....	5
3.2. LSB Implementation Conformance .....	5
3.3. LSB Application Conformance .....	6
4. Definitions .....	7
5. Terminology .....	8
6. Documentation Conventions .....	9

# List of Tables

2-1. Normative References .....	2
3-1. Standard Library Names .....	5

# Foreword

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

# Introduction

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

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



# **I. Introductory Elements**



# Chapter 1. Scope

## 1.1. General

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

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

9 The LSB-generic document shall be used in conjunction with an architecture-specific supplement. Whenever a section  
10 of the LSB-generic specification shall be supplemented by architecture-specific information, the LSB-generic  
11 document includes a reference to the architecture supplement. Architecture supplements may also contain additional  
12 information that is not referenced in the LSB-generic document.

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

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

## 1.2. Module Specific Scope

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

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

## Chapter 2. Normative References

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

4 **Table 2-1. Normative References**

Name	Title	URL
DWARF Debugging Information Format	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	<a href="http://www.eagercon.com/dwarf/dwarf-2.0.0.pdf">http://www.eagercon.com/dwarf/dwarf-2.0.0.pdf</a>
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	<a href="http://www.pathname.com/fhs/">http://www.pathname.com/fhs/</a>
IEEE Std 754-1985	IEEE Standard 754 for Binary Floating-Point Arithmetic	<a href="http://www.ieee.org/">http://www.ieee.org/</a>
Intel® Architecture Software Developer's Manual Volume 3	The IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide	<a href="http://developer.intel.com/design/pentium4/manuals/245472.htm">http://developer.intel.com/design/pentium4/manuals/245472.htm</a>
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	<a href="http://www.unix.org/version3/">http://www.unix.org/version3/</a>
Large File Support	Large File Support	<a href="http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html">http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html</a>

Name	Title	URL
Li18nux Globalization Specification	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	<a href="http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm">http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm</a>
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	<a href="http://www.lanana.org/docs/device-list/devices.txt">http://www.lanana.org/docs/device-list/devices.txt</a>
PAM	Open Software Foundation, Request For Comments: 86.0 , October 1995, V. Samar & R.Schemers (SunSoft)	<a href="http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt">http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt</a>
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	<a href="http://www.ietf.org/rfc/rfc1321.txt">http://www.ietf.org/rfc/rfc1321.txt</a>
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	<a href="http://www.ietf.org/rfc/rfc1833.txt">http://www.ietf.org/rfc/rfc1833.txt</a>
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	<a href="http://www.ietf.org/rfc/rfc1951.txt">http://www.ietf.org/rfc/rfc1951.txt</a>
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	<a href="http://www.ietf.org/rfc/rfc1952.txt">http://www.ietf.org/rfc/rfc1952.txt</a>
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	<a href="http://www.ietf.org/rfc/rfc2440.txt">http://www.ietf.org/rfc/rfc2440.txt</a>
SUSv2	CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)	<a href="http://www.opengroup.org/publications/catalog/un.htm">http://www.opengroup.org/publications/catalog/un.htm</a>
SUSv2 Command and Utilities	The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	<a href="http://www.opengroup.org/publications/catalog/un.htm">http://www.opengroup.org/publications/catalog/un.htm</a>
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	<a href="http://www.caldera.com/developers/devspecs/gabi41.pdf">http://www.caldera.com/developers/devspecs/gabi41.pdf</a>
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December	<a href="http://www.caldera.com/developers">http://www.caldera.com/developers</a>

Name	Title	URL
	2003	<a href="/gabi/2003-12-17/contents.html">/gabi/2003-12-17/contents.html</a>
System V ABI, IA32 Supplement	System V Application Binary Interface - Intel386™ Architecture Processor Supplement, Fourth Edition	<a href="http://www.caldera.com/developers/devspecs/abi386-4.pdf">http://www.caldera.com/developers/devspecs/abi386-4.pdf</a>
The Intel® Architecture Software Developer's Manual Volume 1	The IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture	<a href="http://developer.intel.com/design/pentium4/manuals/245470.htm">http://developer.intel.com/design/pentium4/manuals/245470.htm</a>
The Intel® Architecture Software Developer's Manual Volume 2	The IA-32 Intel® Architecture Software Developer's Manual Volume 2: Instruction Set Reference	<a href="http://developer.intel.com/design/pentium4/manuals/245471.htm">http://developer.intel.com/design/pentium4/manuals/245471.htm</a>
this specification	Linux Standard Base	<a href="http://www.linuxbase.org/spec/">http://www.linuxbase.org/spec/</a>
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	<a href="http://www.opengroup.org/publications/catalog/un.htm">http://www.opengroup.org/publications/catalog/un.htm</a>
zlib Manual	zlib 1.2 Manual	<a href="http://www.gzip.org/zlib/">http://www.gzip.org/zlib/</a>

# Chapter 3. Requirements

## 3.1. Relevant Libraries

1 The libraries listed in Table 3-1 shall be available on IA32 Linux Standard Base systems, with the specified runtime  
2 names. These names override or supplement the names specified in the generic LSB specification. The specified  
3 program interpreter, referred to as proginterp in this table, shall be used to load the shared libraries specified by  
4 DT\_NEEDED entries at run time.

5 **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

6  
7 These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

## 3.2. LSB Implementation Conformance

8 A conforming implementation shall satisfy the following requirements:

- 9 • The implementation shall implement fully the architecture described in the hardware manual for the target  
10 processor architecture.
- 11 • The implementation shall be capable of executing compiled applications having the format and using the system  
12 interfaces described in this document.
- 13 • The implementation shall provide libraries containing the interfaces specified by this document, and shall provide a  
14 dynamic linking mechanism that allows these interfaces to be attached to applications at runtime. All the interfaces  
15 shall behave as specified in this document.
- 16 • The map of virtual memory provided by the implementation shall conform to the requirements of this document.
- 17 • The implementation's low-level behavior with respect to function call linkage, system traps, signals, and other such  
18 activities shall conform to the formats described in this document.

- 19 • The implementation shall provide all of the mandatory interfaces in their entirety.
- 20 • The implementation may provide one or more of the optional interfaces. Each optional interface that is provided
- 21 shall be provided in its entirety. The product documentation shall state which optional interfaces are provided.
- 22 • The implementation shall provide all files and utilities specified as part of this document in the format defined here
- 23 and in other referenced documents. All commands and utilities shall behave as required by this document. The
- 24 implementation shall also provide all mandatory components of an application's runtime environment that are
- 25 included or referenced in this document.
- 26 • The implementation, when provided with standard data formats and values at a named interface, shall provide the
- 27 behavior defined for those values and data formats at that interface. However, a conforming implementation may
- 28 consist of components which are separately packaged and/or sold. For example, a vendor of a conforming
- 29 implementation might sell the hardware, operating system, and windowing system as separately packaged items.
- 30 • The implementation may provide additional interfaces with different names. It may also provide additional
- 31 behavior corresponding to data values outside the standard ranges, for standard named interfaces.

### 3.3. LSB Application Conformance

32 A conforming application shall satisfy the following requirements:

- 33 • Its executable files are either shell scripts or object files in the format defined for the Object File Format system
- 34 interface.
- 35 • Its object files participate in dynamic linking as defined in the Program Loading and Linking System interface.
- 36 • It employs only the instructions, traps, and other low-level facilities defined in the Low-Level System interface as
- 37 being for use by applications.
- 38 • If it requires any optional interface defined in this document in order to be installed or to execute successfully, the
- 39 requirement for that optional interface is stated in the application's documentation.
- 40 • It does not use any interface or data format that is not required to be provided by a conforming implementation,
- 41 unless:
  - 42 • If such an interface or data format is supplied by another application through direct invocation of that application
  - 43 during execution, that application is in turn an LSB conforming application.
  - 44 • The use of that interface or data format, as well as its source, is identified in the documentation of the application.
- 45 • It shall not use any values for a named interface that are reserved for vendor extensions.

46 A strictly conforming application does not require or use any interface, facility, or implementation-defined extension

47 that is not defined in this document in order to be installed or to execute successfully.



## Chapter 4. Definitions

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

3 can

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

5 cannot

6 be unable to; there is no possibility of; it is not possible to

7 may

8 is permitted; is allowed; is permissible

9 need not

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

11 shall

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

13 shall not

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

15 should

16 it is recommended that; ought to

17 should not

18 it is not recommended that; ought not to

# Chapter 5. Terminology

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

2 archLSB

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

5 Binary Standard

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

7 gLSB

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

10 implementation-defined

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

16 Shell Script

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

19 Source Standard

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

21 undefined

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

27 unspecified

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

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

## Chapter 6. Documentation Conventions

1 Throughout this document, the following typographic conventions are used:

2 `function()`

3 the name of a function

4 **command**

5 the name of a command or utility

6 `CONSTANT`

7 a constant value

8 *parameter*

9 a parameter

10 `variable`

11 a variable

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

14 `name`

15 the name of the interface

16 `(symver)`

17 An optional symbol version identifier, if required.

18 `[refno]`

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

20 For example,

21 

<code>forkpty(GLIBC_2.0) [1]</code>
-------------------------------------

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

# **ELF Specification**

2

3 **ELF Specification**

# Table of Contents

<b>I. Low Level System Information</b> .....	<b>15</b>
1. Machine Interface.....	1
1.1. Processor Architecture.....	1
1.2. Data Representation.....	1
1.2.1. Byte Ordering.....	1
1.2.2. Fundamental Types.....	1
1.2.3. Aggregates and Unions.....	2
1.2.4. Bit Fields.....	2
2. Function Calling Sequence.....	3
2.1. CPU Registers.....	3
2.2. Floating Point Registers.....	3
2.3. Stack Frame.....	3
2.4. Arguments.....	3
2.4.1. Integral/Pointer.....	3
2.4.2. Floating Point.....	3
2.4.3. Struct and Union Point.....	3
2.4.4. Variable Arguments.....	3
2.5. Return Values.....	3
2.5.1. Void.....	3
2.5.2. Integral/Pointer.....	4
2.5.3. Floating Point.....	4
2.5.4. Struct and Union Point.....	4
3. Operating System Interface.....	5
3.1. Virtual Address Space.....	5
3.1.1. Page Size.....	5
3.1.2. Virtual Address Assignments.....	5
3.1.3. Managing the PProcess Stack.....	5
3.1.4. Coding Guidelines.....	5
3.2. Processor Execution Mode.....	5
3.3. Exception Interface.....	5
3.3.1. Hardware Exception Types.....	5
3.3.2. Software Trap Types.....	5
3.4. Signal Delivery.....	5
3.4.1. Signal Handler Interface.....	6
4. Process Initialization.....	7
4.1. Special Registers.....	7
4.2. Process Stack (on entry).....	7
4.3. Auxilliary Vectors.....	7
4.4. Environment.....	7
5. Coding Examples.....	8
5.1. Code Model Overview/Architecture Constraints.....	8
5.2. Position-Independent Function Prologue.....	8
5.3. Data Objects.....	8

5.3.1. Absolute Load & Store.....	8
5.3.2. Position Relative Load & Store.....	8
5.4. Function Calls.....	8
5.4.1. Absolute Direct Function Call.....	8
5.4.2. Absolute Indirect Function Call.....	8
5.4.3. Position-Independent Direct Function Call.....	8
5.4.4. Position-Independent Indirect Function Call.....	8
5.5. Branching.....	9
5.5.1. Branch Instruction.....	9
5.5.2. Absolute switch() code.....	9
5.5.3. Position-Independent switch() code.....	9
6. C Stack Frame.....	10
6.1. Variable Argument List.....	10
6.2. Dynamic Allocation of Stack Space.....	10
7. Debug Information.....	11
<b>II. Object Format.....</b>	<b>12</b>
8. ELF Header.....	13
8.1. Machine Information.....	13
8.1.1. File Class.....	13
8.1.2. Data Encoding.....	13
8.1.3. OS Identification.....	13
8.1.4. Processor Identification.....	13
8.1.5. Processor Specific Flags.....	13
9. Special Sections.....	14
9.1. Special Sections.....	14
9.1.1. ELF Special Sections.....	14
9.1.2. Addition Special Sections.....	14
10. Symbol Table.....	15
11. Relocation.....	16
11.1. Relocation Types.....	16
<b>III. Program Loading and Dynamic Linking.....</b>	<b>17</b>
12. Program Header.....	18
12.1. Types.....	18
12.2. Flags.....	18
13. Program Loading.....	19
14. Dynamic Linking.....	20
14.1. Dynamic Section.....	20
14.2. Global Offset Table.....	20
14.3. Shared Object Dependencies.....	20
14.4. Function Addresses.....	20
14.5. Procedure Linkage Table.....	20
14.6. Initialization and Termination Functions.....	20

# List of Tables

1-1. Scalar Types .....	1
9-1. ELF Special Sections.....	14
9-2. Additional Special Sections.....	14



# **I. Low Level System Information**



# Chapter 1. Machine Interface

## 1.1. Processor Architecture

1 The IA32 Architecture is specified by the following documents

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

5 Only the features of the Intel486 processor instruction set may be assumed to be present. An application is responsible  
6 for determining if any additional instruction set features are available before using those additional features. If a  
7 feature is not present, then the application may not use it.

8 Only instructions which do not require elevated privileges may be used.

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

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

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

## 1.2. Data Representation

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

### 1.2.1. Byte Ordering

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

### 1.2.2. Fundamental Types

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

20 **Table 1-1. Scalar Types**

Type	C	sizeof	Alignment (bytes)	Intel386 Archi- tecture
Integral	long long	8	4	signed double word
	signed long long			
	unsigned long long	8	4	unsigned double

21

Type	C	sizeof	Alignment (bytes)	Intel386 Architecture
				word

### 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

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

## 2.1. CPU Registers

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

## 2.2. Floating Point Registers

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

## 2.3. Stack Frame

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

## 2.4. Arguments

### 2.4.1. Integral/Pointer

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

### 2.4.2. Floating Point

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

### 2.4.3. Struct and Union Point

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

### 2.4.4. Variable Arguments

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

## 2.5. Return Values

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

### 2.5.1. Void

11 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

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

## 3.1. Virtual Address Space

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

### 3.1.1. Page Size

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

### 3.1.2. Virtual Address Assignments

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

### 3.1.3. Managing the PProcess Stack

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

### 3.1.4. Coding Guidelines

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

## 3.2. Processor Execution Mode

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

## 3.3. Exception Interface

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

### 3.3.1. Hardware Exception Types

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

### 3.3.2. Software Trap Types

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

## 3.4. Signal Delivery

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

### **3.4.1. Signal Handler Interface**

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



# Chapter 4. Process Initialization

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

## 4.1. Special Registers

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

## 4.2. Process Stack (on entry)

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

## 4.3. Auxilliary Vectors

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

## 4.4. Environment

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

# Chapter 5. Coding Examples

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

## 5.1. Code Model Overview/Architecture Constraints

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

## 5.2. Position-Independent Function Prologue

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

## 5.3. Data Objects

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

### 5.3.1. Absolute Load & Store

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

### 5.3.2. Position Relative Load & Store

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

## 5.4. Function Calls

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

### 5.4.1. Absolute Direct Function Call

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

### 5.4.2. Absolute Indirect Function Call

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

### 5.4.3. Position-Independent Direct Function Call

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

### 5.4.4. Position-Independent Indirect Function Call

12 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

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

# Chapter 8. ELF Header

## 8.1. Machine Information

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

### 8.1.1. File Class

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

### 8.1.2. Data Encoding

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

### 8.1.3. OS Identification

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

### 8.1.4. Processor Identification

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

### 8.1.5. Processor Specific Flags

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

# Chapter 9. Special Sections

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

## 9.1. Special Sections

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

### 9.1.1. ELF Special Sections

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

5 **Table 9-1. ELF Special Sections**

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

7 .got

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

10 .plt

11 This section holds the procedure linkage table.

### 9.1.2. Additional Special Sections

12 The following additional sections are defined here.

13 **Table 9-2. Additional Special Sections**

Name	Type	Attributes
.rel.dyn	SHT_REL	SHF_ALLOC

15 .rel.dyn

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



## Chapter 10. Symbol Table

- 1 LSB-conforming applications shall use the Symbol Table as defined in Chapter 4 of the System V ABI, IA32
- 2 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**

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

# Chapter 12. Program Header

<sup>1</sup> 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

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

## 14.1. Dynamic Section

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

3 DT\_PLTGOT

4 On the Intel386 architecture, this entry's `d_ptr` member gives the address of the first entry in the global offset  
5 table.

## 14.2. Global Offset Table

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

## 14.3. Shared Object Dependencies

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

## 14.4. Function Addresses

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

## 14.5. Procedure Linkage Table

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

## 14.6. Initialization and Termination Functions

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

# **Linux Standard Base Specification**

2

3 **Linux Standard Base Specification**



# Table of Contents

<b>I. Base Libraries</b> .....	<b>27</b>
1. Libraries .....	1
1.1. Program Interpreter/Dynamic Linker .....	1
1.2. Interfaces for libc .....	1
1.2.1. RPC .....	1
1.2.1.1. Interfaces for RPC .....	1
1.2.2. System Calls .....	2
1.2.2.1. Interfaces for System Calls .....	2
1.2.3. Standard I/O .....	4
1.2.3.1. Interfaces for Standard I/O .....	4
1.2.4. Signal Handling .....	5
1.2.4.1. Interfaces for Signal Handling .....	5
1.2.5. Localization Functions .....	6
1.2.5.1. Interfaces for Localization Functions .....	6
1.2.6. Socket Interface .....	7
1.2.6.1. Interfaces for Socket Interface .....	7
1.2.7. Wide Characters .....	8
1.2.7.1. Interfaces for Wide Characters .....	8
1.2.8. String Functions .....	9
1.2.8.1. Interfaces for String Functions .....	9
1.2.9. IPC Functions .....	10
1.2.9.1. Interfaces for IPC Functions .....	10
1.2.10. Regular Expressions .....	11
1.2.10.1. Interfaces for Regular Expressions .....	11
1.2.11. Character Type Functions .....	11
1.2.11.1. Interfaces for Character Type Functions .....	11
1.2.12. Time Manipulation .....	12
1.2.12.1. Interfaces for Time Manipulation .....	12
1.2.13. Terminal Interface Functions .....	13
1.2.13.1. Interfaces for Terminal Interface Functions .....	13
1.2.14. System Database Interface .....	14
1.2.14.1. Interfaces for System Database Interface .....	14
1.2.15. Language Support .....	14
1.2.15.1. Interfaces for Language Support .....	14
1.2.16. Large File Support .....	15
1.2.16.1. Interfaces for Large File Support .....	15
1.2.17. Standard Library .....	15
1.2.17.1. Interfaces for Standard Library .....	15
1.3. Data Definitions for libc .....	17
1.3.1. errno.h .....	18
1.3.2. inttypes.h .....	18
1.3.3. limits.h .....	18
1.3.4. setjmp.h .....	18

1.3.5. signal.h .....	18
1.3.6. stddef.h .....	20
1.3.7. sys/ioctl.h .....	20
1.3.8. sys/ipc.h .....	20
1.3.9. sys/mman.h .....	21
1.3.10. sys/msg.h .....	21
1.3.11. sys/sem.h .....	21
1.3.12. sys/shm.h .....	21
1.3.13. sys/socket.h .....	22
1.3.14. sys/stat.h .....	22
1.3.15. sys/statvfs.h .....	23
1.3.16. sys/types.h .....	24
1.3.17. termios.h .....	24
1.3.18. ucontext.h .....	25
1.3.19. unistd.h .....	26
1.3.20. utmp.h .....	26
1.3.21. utmpx.h .....	27
1.4. Interfaces for libm .....	27
1.4.1. Math .....	27
1.4.1.1. Interfaces for Math .....	27
1.5. Interfaces for libpthread .....	31
1.5.1. Realtime Threads .....	31
1.5.1.1. Interfaces for Realtime Threads .....	31
1.5.2. Advanced Realtime Threads .....	31
1.5.2.1. Interfaces for Advanced Realtime Threads .....	31
1.5.3. Posix Threads .....	32
1.5.3.1. Interfaces for Posix Threads .....	32
1.6. Interfaces for libgcc_s .....	33
1.6.1. Unwind Library .....	33
1.6.1.1. Interfaces for Unwind Library .....	33
1.7. Interface Definitions for libgcc_s .....	34
_Unwind_DeleteException .....	34
_Unwind_Find_FDE .....	35
_Unwind_ForcedUnwind .....	36
_Unwind_GetDataRelBase .....	37
_Unwind_GetGR .....	37
_Unwind_GetIP .....	37
_Unwind_GetLanguageSpecificData .....	38
_Unwind_GetRegionStart .....	38
_Unwind_GetTextRelBase .....	38
_Unwind_RaiseException .....	39
_Unwind_Resume .....	40
_Unwind_SetGR .....	40
_Unwind_SetIP .....	40
1.8. Interfaces for libdl .....	40
1.8.1. Dynamic Loader .....	41
1.8.1.1. Interfaces for Dynamic Loader .....	41
1.9. Interfaces for libcrypt .....	41

1.9.1. Encryption .....	41
1.9.1.1. Interfaces for Encryption .....	41
<b>II. Utility Libraries .....</b>	<b>43</b>
2. Libraries .....	44
2.1. Interfaces for libz .....	44
2.1.1. Compression Library .....	44
2.1.1.1. Interfaces for Compression Library .....	44
2.2. Interfaces for libncurses .....	44
2.2.1. Curses .....	44
2.2.1.1. Interfaces for Curses .....	44
2.3. Interfaces for libutil .....	44
2.3.1. Utility Functions .....	45
2.3.1.1. Interfaces for Utility Functions .....	45
<b>A. Alphabetical Listing of Interfaces .....</b>	<b>46</b>
A.1. libgcc_s .....	46

# List of Tables

1-1. libc Definition.....	1
1-2. libc - RPC Function Interfaces .....	1
1-3. libc - System Calls Function Interfaces .....	2
1-4. libc - Standard I/O Function Interfaces .....	4
1-5. libc - Standard I/O Data Interfaces .....	5
1-6. libc - Signal Handling Function Interfaces .....	5
1-7. libc - Signal Handling Data Interfaces.....	6
1-8. libc - Localization Functions Function Interfaces .....	6
1-9. libc - Localization Functions Data Interfaces .....	7
1-10. libc - Socket Interface Function Interfaces .....	7
1-11. libc - Socket Interface Deprecated Function Interfaces .....	8
1-12. libc - Wide Characters Function Interfaces .....	8
1-13. libc - String Functions Function Interfaces.....	9
1-14. libc - IPC Functions Function Interfaces .....	10
1-15. libc - Regular Expressions Function Interfaces .....	11
1-16. libc - Regular Expressions Deprecated Function Interfaces .....	11
1-17. libc - Regular Expressions Deprecated Data Interfaces.....	11
1-18. libc - Character Type Functions Function Interfaces.....	12
1-19. libc - Time Manipulation Function Interfaces .....	12
1-20. libc - Time Manipulation Deprecated Function Interfaces .....	13
1-21. libc - Time Manipulation Data Interfaces.....	13
1-22. libc - Terminal Interface Functions Function Interfaces.....	13
1-23. libc - System Database Interface Function Interfaces.....	14
1-24. libc - Language Support Function Interfaces.....	14
1-25. libc - Large File Support Function Interfaces .....	15
1-26. libc - Standard Library Function Interfaces .....	15
1-27. libc - Standard Library Data Interfaces .....	17
1-28. libm Definition .....	27
1-29. libm - Math Function Interfaces .....	27
1-30. libm - Math Data Interfaces.....	31
1-31. libpthread Definition .....	31
1-32. libpthread - Posix Threads Function Interfaces .....	32
1-33. libgcc_s Definition .....	33
1-34. libgcc_s - Unwind Library Function Interfaces .....	33
1-35. libdl Definition .....	41
1-36. libdl - Dynamic Loader Function Interfaces .....	41
1-37. libcrypt Definition .....	41
1-38. libcrypt - Encryption Function Interfaces .....	41
2-1. libz Definition.....	44
2-2. libncurses Definition .....	44
2-3. libutil Definition .....	44
2-4. libutil - Utility Functions Function Interfaces .....	45
A-1. libgcc_s Function Interfaces .....	46

# I. Base Libraries



# Chapter 1. Libraries

- 1 An LSB-conforming implementation shall support some base libraries which provide interfaces for accessing the  
2 operating system, processor and other hardware in the system.
- 3 Interfaces that are unique to the IA32 platform are defined here. This section should be used in conjunction with the  
4 corresponding section in the Linux Standard Base Specification.

## 1.1. Program Interpreter/Dynamic Linker

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

## 1.2. Interfaces for libc

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

7 **Table 1-1. libc Definition**

Library:	libc
SONAME:	libc.so.6

- 9 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  
10 SVID Issue 4

### 1.2.1. RPC

#### 11 1.2.1.1. Interfaces for RPC

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

14 **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_spcrerror(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]	

15

16 *Referenced Specification(s)*

17 [1]. SVID Issue 4

18 [2]. this specification

19 [3]. SVID Issue 3

## 1.2.2. System Calls

### 1.2.2.1. Interfaces for System Calls

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

23 **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_2.0) [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]



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

24

25 *Referenced Specification(s)*

26 [1]. this specification

27 [2]. ISO POSIX (2003)

28 [3]. Large File Support

29 [4]. SUSv2

## 1.2.3. Standard I/O

### 1.2.3.1. Interfaces for Standard I/O

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

33 **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(G LIBC_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]	vsprintf(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(GL IBC_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(GLI BC_2.0) [2]	sprintf(GLIBC_2.0) [2]	

34

35 *Referenced Specification(s)*

36 [1]. this specification

37 [2]. ISO POSIX (2003)

38 [3]. SUSv2

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

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

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

42

43 *Referenced Specification(s)*

44 [1]. ISO POSIX (2003)

## 1.2.4. Signal Handling

### 1.2.4.1. Interfaces for Signal Handling

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

48 **Table 1-6. libc - Signal Handling Function Interfaces**

__libc_current_sigrt max(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]	

49

50 *Referenced Specification(s)*

51 [1]. this specification

52 [2]. ISO POSIX (2003)

53 [3]. SUSv2

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

56 **Table 1-7. libc - Signal Handling Data Interfaces**

__sys_siglist(GLIBC_2.3.3) [1]				
--------------------------------	--	--	--	--

57

58 *Referenced Specification(s)*

59 [1]. this specification

## 1.2.5. Localization Functions

### 1.2.5.1. Interfaces for Localization Functions

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

63 **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(GLIBC_2.0) [1]	dcgettext(GLIBC_2.0) [1]	gettext(GLIBC_2.0) [1]	localeconv(GLIBC_2.0) [1]	textdomain(GLIBC_2.0) [1]

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]	

64

65 *Referenced Specification(s)*

66 [1]. this specification

67 [2]. ISO POSIX (2003)

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

70 **Table 1-9. libc - Localization Functions Data Interfaces**

_nl_msg_cat_cntr(GLIBC_2.0) [1]				
---------------------------------	--	--	--	--

71

72 *Referenced Specification(s)*

73 [1]. this specification

## 1.2.6. Socket Interface

### 1.2.6.1. Interfaces for Socket Interface

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

77 **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]	

78

79 *Referenced Specification(s)*

80 [1]. this specification

81 [2]. ISO POSIX (2003)

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

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

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

gethostbyname_r(G LIBC_2.1.2) [1]				
--------------------------------------	--	--	--	--

87  
 88 *Referenced Specification(s)*

89 [1]. this specification

## 1.2.7. Wide Characters

### 1.2.7.1. Interfaces for Wide Characters

90  
 91 An LSB conforming implementation shall provide the architecture specific functions for Wide Characters specified in  
 92 Table 1-12, with the full functionality as described in the referenced underlying specification.

93 **Table 1-12. libc - Wide Characters Function Interfaces**

__wctod_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]
__wctof_internal( GLIBC_2.0) [1]	mbsnrto wcs(GLIBC _2.0) [1]	wcpcpy(GLIBC_2.0 ) [1]	wcsnrto mbs(GLIBC _2.0) [1]	wcstouq(GLIBC_2. 0) [1]
__wctol_internal(G LIBC_2.0) [1]	mbsrtowcs(GLIBC_ 2.0) [2]	wcpncpy(GLIBC_2. 0) [1]	wcsprk(GLIBC_2. 0) [2]	wcswcs(GLIBC_2.1 ) [2]
__wctold_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]
__wctoul_internal( GLIBC_2.0) [1]	mbtowc(GLIBC_2. 0) [2]	wcscasecmp(GLIB C_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]	wctod(GLIBC_2.0) [2]	wctrans(GLIBC_2.0 ) [2]
fputwc(GLIBC_2.2) [2]	swscanf(GLIBC_2. 2) [2]	wscoll(GLIBC_2.0 ) [2]	wctof(GLIBC_2.0) [2]	wctype(GLIBC_2.0 ) [2]
fputws(GLIBC_2.2) [2]	towctrans(GLIBC_2 .0) [2]	wcscpy(GLIBC_2.0 ) [2]	wctoimax(GLIBC_ 2.1) [2]	wcwidth(GLIBC_2. 0) [2]

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

94

95 *Referenced Specification(s)*

96 [1]. this specification

97 [2]. ISO POSIX (2003)

## 1.2.8. String Functions

### 1.2.8.1. Interfaces for String Functions

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

101 **Table 1-13. libc - String Functions Function Interfaces**

<code>__memcpy</code> (GLIBC_2.0) [1]	<code>bzero</code> (GLIBC_2.0) [2]	<code>strcasestr</code> (GLIBC_2.1) [1]	<code>strncasecmp</code> (GLIBC_2.0) [2]	<code>strtoimax</code> (GLIBC_2.1) [2]
<code>__rawmemchr</code> (GLIBC_2.1) [1]	<code>ffs</code> (GLIBC_2.0) [2]	<code>strcat</code> (GLIBC_2.0) [2]	<code>strncat</code> (GLIBC_2.0) [2]	<code>strtok</code> (GLIBC_2.0) [2]
<code>__stpcpy</code> (GLIBC_2.0) [1]	<code>index</code> (GLIBC_2.0) [2]	<code>strchr</code> (GLIBC_2.0) [2]	<code>strncmp</code> (GLIBC_2.0) [2]	<code>strtok_r</code> (GLIBC_2.0) [2]
<code>__strdup</code> (GLIBC_2.0) [1]	<code>memcpy</code> (GLIBC_2.0) [2]	<code>strcmp</code> (GLIBC_2.0) [2]	<code>strncpy</code> (GLIBC_2.0) [2]	<code>strtold</code> (GLIBC_2.0) [2]
<code>__strtod_internal</code> (GLIBC_2.0) [1]	<code>memchr</code> (GLIBC_2.0) [2]	<code>strcoll</code> (GLIBC_2.0) [2]	<code>strndup</code> (GLIBC_2.0) [1]	<code>strtoll</code> (GLIBC_2.0) [2]
<code>__strtof_internal</code> (GLIBC_2.0) [1]	<code>memcmp</code> (GLIBC_2.0) [2]	<code>strcpy</code> (GLIBC_2.0) [2]	<code>strlen</code> (GLIBC_2.0) [2]	<code>strtoq</code> (GLIBC_2.0) [2]

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]	strchr(GLIBC_2.0) [2]	strtouq(GLIBC_2.0) [1]
__strtol_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]	strcasestr(GLIBC_2.0) [2]	strlen(GLIBC_2.0) [2]	strtof(GLIBC_2.0) [2]	

102

103 *Referenced Specification(s)*

104 [1]. this specification

105 [2]. ISO POSIX (2003)

## 1.2.9. IPC Functions

### 1.2.9.1. Interfaces for IPC Functions

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

109 **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]	

110

111 *Referenced Specification(s)*

112 [1]. ISO POSIX (2003)



## 1.2.10. Regular Expressions

### 1.2.10.1. Interfaces for Regular Expressions

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

116 **Table 1-15. libc - Regular Expressions Function Interfaces**

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

118 *Referenced Specification(s)*

119 [1]. ISO POSIX (2003)

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

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

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

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

126 *Referenced Specification(s)*

127 [1]. SUSv2

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

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

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

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

134 *Referenced Specification(s)*

135 [1]. SUSv2

## 1.2.11. Character Type Functions

### 1.2.11.1. Interfaces for Character Type Functions

136 An LSB conforming implementation shall provide the architecture specific functions for Character Type Functions  
137 specified in Table 1-18, with the full functionality as described in the referenced underlying specification.  
138

139 **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]	iswalphabet(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_2.0) [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]	
isctrl(GLIBC_2.0) [2]	isupper(GLIBC_2.0) [2]	iswgraph(GLIBC_2.0) [2]	isxdigit(GLIBC_2.0) [2]	

140

141 *Referenced Specification(s)*

142 [1]. this specification

143 [2]. ISO POSIX (2003)

## 1.2.12. Time Manipulation

### 1.2.12.1. Interfaces for Time Manipulation

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

147 **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]	alarm(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]	

148

149 *Referenced Specification(s)*

150 [1]. this specification

151 [2]. ISO POSIX (2003)

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

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

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

adjtimex(GLIBC_2.0) [1]				
-------------------------	--	--	--	--

158  
 159 *Referenced Specification(s)*

160 [1]. this specification

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

163 **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]		

164  
 165 *Referenced Specification(s)*

166 [1]. this specification

167 [2]. ISO POSIX (2003)

## 1.2.13. Terminal Interface Functions

### 1.2.13.1. Interfaces for Terminal Interface Functions

168 An LSB conforming implementation shall provide the architecture specific functions for Terminal Interface Functions  
 169 specified in Table 1-22, with the full functionality as described in the referenced underlying specification.  
 170

171 **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]

172  
 173 *Referenced Specification(s)*

174 [1]. ISO POSIX (2003)

175 [2]. this specification

## 1.2.14. System Database Interface

### 1.2.14.1. Interfaces for System Database Interface

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

179 **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]

180

181 *Referenced Specification(s)*

182 [1]. ISO POSIX (2003)

183 [2]. this specification

184 [3]. SUSv2

## 1.2.15. Language Support

### 1.2.15.1. Interfaces for Language Support

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

188 **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]	
----------------------------------	-------------------------------	----------------------------------	-----------------------------	--

189

190 *Referenced Specification(s)*

191 [1]. this specification

## 1.2.16. Large File Support

### 1.2.16.1. Interfaces for Large File Support

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

195 **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]	

196

197 *Referenced Specification(s)*

198 [1]. this specification

199 [2]. Large File Support

## 1.2.17. Standard Library

### 1.2.17.1. Interfaces for Standard Library

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

203 **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(GLIB C_2.0) [2]	div(GLIBC_ 2.0) [1]	glob64(GLIBC_ 2.2) [2]	makecontext(GLIB C_2.1) [1]	srand48(GLIBC_ 2.0) [1]
__cxa_atexit(GLIB C_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(G LIBC_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]	mrnd48(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]	nrnd48(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]
__isnani(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]	ttynam(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]	ttynam_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_2.0)	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]	

204

205 *Referenced Specification(s)*

206 [1]. ISO POSIX (2003)

207 [2]. this specification

208 [3]. SUSv2

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

211 **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]	

212

213 *Referenced Specification(s)*

214 [1]. this specification

215 [2]. ISO POSIX (2003)

### 1.3. Data Definitions for libc

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

219 These definitions are intended to supplement those provided in the referenced underlying specifications.

220 This specification uses ISO/IEC 9899 C Language as the reference programming language, and data definitions are  
 221 specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of  
 222 these data objects does not preclude their use by other programming languages.

### 1.3.1. errno.h

```
223
224 #define EDEADLOCK          EDEADLK
```

### 1.3.2. inttypes.h

```
225
226 typedef long long intmax_t;
227 typedef unsigned int uintptr_t;
228 typedef unsigned long long uintmax_t;
229 typedef unsigned long long uint64_t;
```

### 1.3.3. limits.h

```
230
231 #define LONG_MAX           0x7FFFFFFFL
232 #define ULONG_MAX         0xFFFFFFFFUL
233
234 #define CHAR_MAX          SCHAR_MAX
235 #define CHAR_MIN          SCHAR_MIN
```

### 1.3.4. setjmp.h

```
236
237 typedef int __jmp_buf[6];
```

### 1.3.5. signal.h

```
238
239 struct sigaction
240 {
241     union
242     {
243         sighandler_t _sa_handler;
244         void (*_sa_sigaction) (int, siginfo_t *, void *);
245     }
246     __sigaction_handler;
247     sigset_t sa_mask;
248     unsigned long sa_flags;
249     void (*sa_restorer) (void);
250 }
251 ;
252 #define MINSIGSTKSZ      2048
253 #define SIGSTKSZ         8192
254
255 struct _fpreg
```



```

256 {
257     unsigned short significand[4];
258     unsigned short exponent;
259 }
260 ;
261 struct _fpxreg
262 {
263     unsigned short significand[4];
264     unsigned short exponent;
265     unsigned short padding[3];
266 }
267 ;
268 struct _xmmreg
269 {
270     unsigned long element[4];
271 }
272 ;
273
274 struct _fpstate
275 {
276     unsigned long cw;
277     unsigned long sw;
278     unsigned long tag;
279     unsigned long ipoff;
280     unsigned long cssel;
281     unsigned long dataoff;
282     unsigned long datasel;
283     struct _fpreg _st[8];
284     unsigned short status;
285     unsigned short magic;
286     unsigned long _fxsr_env[6];
287     unsigned long mxcsr;
288     unsigned long reserved;
289     struct _fpxreg _fxsr_st[8];
290     struct _xmmreg _xmm[8];
291     unsigned long padding[56];
292 }
293 ;
294
295 struct sigcontext
296 {
297     unsigned short gs;
298     unsigned short __gsh;
299     unsigned short fs;
300     unsigned short __fsh;
301     unsigned short es;
302     unsigned short __esh;
303     unsigned short ds;
304     unsigned short __dsh;
305     unsigned long edi;
306     unsigned long esi;
307     unsigned long ebp;
308     unsigned long esp;

```

```

309     unsigned long ebx;
310     unsigned long edx;
311     unsigned long ecx;
312     unsigned long eax;
313     unsigned long trapno;
314     unsigned long err;
315     unsigned long eip;
316     unsigned short cs;
317     unsigned short __csh;
318     unsigned long eflags;
319     unsigned long esp_at_signal;
320     unsigned short ss;
321     unsigned short __ssh;
322     struct _fpstate *fpstate;
323     unsigned long oldmask;
324     unsigned long cr2;
325 }
326 ;

```

### 1.3.6. stddef.h

```

327
328 typedef unsigned int size_t;
329 typedef int ptrdiff_t;

```

### 1.3.7. sys/ioctl.h

```

330
331 #define FIONREAD          0x541B
332 #define TIOCNOTTY       0x5422

```

### 1.3.8. sys/ipc.h

```

333
334 struct ipc_perm
335 {
336     key_t __key;
337     uid_t uid;
338     gid_t gid;
339     uid_t cuid;
340     gid_t cgid;
341     unsigned short mode;
342     unsigned short __pad1;
343     unsigned short __seq;
344     unsigned short __pad2;
345     unsigned long __unused1;
346     unsigned long __unused2;
347 }
348 ;

```

### 1.3.9. sys/mman.h

```

349
350 #define MCL_CURRENT      1
351 #define MCL_FUTURE      2

```

### 1.3.10. sys/msg.h

```

352
353 typedef unsigned long msgqnum_t;
354 typedef unsigned long msglen_t;
355
356 struct msqid_ds
357 {
358     struct ipc_perm msg_perm;
359     time_t msg_stime;
360     unsigned long __unused1;
361     time_t msg_rtime;
362     unsigned long __unused2;
363     time_t msg_ctime;
364     unsigned long __unused3;
365     unsigned long __msg_cbytes;
366     msgqnum_t msg_qnum;
367     msglen_t msg_qbytes;
368     pid_t msg_lspid;
369     pid_t msg_lrpid;
370     unsigned long __unused4;
371     unsigned long __unused5;
372 }
373 ;

```

### 1.3.11. sys/sem.h

```

374
375 struct semid_ds
376 {
377     struct ipc_perm sem_perm;
378     time_t sem_otime;
379     unsigned long __unused1;
380     time_t sem_ctime;
381     unsigned long __unused2;
382     unsigned long sem_nsems;
383     unsigned long __unused3;
384     unsigned long __unused4;
385 }
386 ;

```

### 1.3.12. sys/shm.h

```

387
388 #define SHMLBA    (__getpagesize())

```

```

389
390 typedef unsigned long shmatt_t;
391
392 struct shmid_ds
393 {
394     struct ipc_perm shm_perm;
395     int shm_segsz;
396     time_t shm_atime;
397     unsigned long __unused1;
398     time_t shm_dtime;
399     unsigned long __unused2;
400     time_t shm_ctime;
401     unsigned long __unused3;
402     pid_t shm_cpid;
403     pid_t shm_lpid;
404     shmatt_t shm_nattch;
405     unsigned long __unused4;
406     unsigned long __unused5;
407 }
408 ;

```

### 1.3.13. sys/socket.h

```

409
410 typedef uint32_t __ss_aligntype;

```

### 1.3.14. sys/stat.h

```

411
412 #define _STAT_VER      3
413
414 struct stat
415 {
416     dev_t st_dev;
417     unsigned short __pad1;
418     unsigned long st_ino;
419     mode_t st_mode;
420     nlink_t st_nlink;
421     pid_t st_uid;
422     gid_t st_gid;
423     dev_t st_rdev;
424     unsigned short __pad2;
425     off_t st_size;
426     blksize_t st_blksize;
427     blkcnt_t st_blocks;
428     struct timespec st_atim;
429     struct timespec st_mtim;
430     struct timespec st_ctim;
431     unsigned long __unused4;
432     unsigned long __unused5;
433 }
434 ;

```

```

435 struct stat64
436 {
437     dev_t st_dev;
438     unsigned int __pad1;
439     ino_t __st_ino;
440     mode_t st_mode;
441     nlink_t st_nlink;
442     uid_t st_uid;
443     gid_t st_gid;
444     dev_t st_rdev;
445     unsigned int __pad2;
446     off64_t st_size;
447     blksize_t st_blksize;
448     blkcnt64_t st_blocks;
449     struct timespec st_atim;
450     struct timespec st_mtim;
451     struct timespec st_ctim;
452     ino64_t st_ino;
453 }
454 ;

```

### 1.3.15. sys/statvfs.h

```

455
456 struct statvfs
457 {
458     unsigned long f_bsize;
459     unsigned long f_frsize;
460     fsblkcnt_t f_blocks;
461     fsblkcnt_t f_bfree;
462     fsblkcnt_t f_bavail;
463     fsfilcnt_t f_files;
464     fsfilcnt_t f_ffree;
465     fsfilcnt_t f_favail;
466     unsigned long f_fsid;
467     int __f_unused;
468     unsigned long f_flag;
469     unsigned long f_namemax;
470     int __f_spare[6];
471 }
472 ;
473 struct statvfs64
474 {
475     unsigned long f_bsize;
476     unsigned long f_frsize;
477     fsblkcnt64_t f_blocks;
478     fsblkcnt64_t f_bfree;
479     fsblkcnt64_t f_bavail;
480     fsfilcnt64_t f_files;
481     fsfilcnt64_t f_ffree;
482     fsfilcnt64_t f_favail;
483     unsigned long f_fsid;

```

```

484     int __f_unused;
485     unsigned long f_flag;
486     unsigned long f_namemax;
487     int __f_spare[6];
488 }
489 ;

```

### 1.3.16. sys/types.h

```

490
491 typedef long long int64_t;
492
493 typedef int32_t ssize_t;

```

### 1.3.17. termios.h

```

494
495 #define OLCUC    0000002
496 #define ONLCR   0000004
497 #define XCASE   0000004
498 #define NLDLY   0000400
499 #define CR1     0001000
500 #define IUCLC   0001000
501 #define CR2     0002000
502 #define CR3     0003000
503 #define CRDLY   0003000
504 #define TAB1    0004000
505 #define TAB2    0010000
506 #define TAB3    0014000
507 #define TABDLY  0014000
508 #define BS1     0020000
509 #define BSDLY   0020000
510 #define VT1     0040000
511 #define VTDLY   0040000
512 #define FF1     0100000
513 #define FFDLY   0100000
514
515 #define VSUSP   10
516 #define VEOL    11
517 #define VREPRINT      12
518 #define VDISCARD     13
519 #define VWERASE      14
520 #define VEOL2       16
521 #define VMIN         6
522 #define VSWTC        7
523 #define VSTART       8
524 #define VSTOP        9
525
526 #define IXON    0002000
527 #define IXOFF   0010000
528
529 #define CS6     0000020

```

```

530 #define CS7      0000040
531 #define CS8      0000060
532 #define CSIZE    0000060
533 #define CSTOPB   0000100
534 #define CREAD    0000200
535 #define PARENB   0000400
536 #define PARODD   0001000
537 #define HUPCL    0002000
538 #define CLOCAL   0004000
539 #define VTIME    5
540
541 #define ISIG      0000001
542 #define ICANON    0000002
543 #define ECHOE     0000020
544 #define ECHOK     0000040
545 #define ECHONL    0000100
546 #define NOFLSH   0000200
547 #define TOSTOP   0000400
548 #define ECHOCTL   0001000
549 #define ECHOPRT  0002000
550 #define ECHOKE    0004000
551 #define FLUSHO    0010000
552 #define PENDIN    0040000
553 #define IEXTEN    0100000

```

### 1.3.18. ucontext.h

```

554
555 typedef int greg_t;
556 #define NGREG     19
557
558 typedef greg_t gregset_t[19];
559
560 struct _libc_fpreg
561 {
562     unsigned short significand[4];
563     unsigned short exponent;
564 }
565 ;
566
567 struct _libc_fpstate
568 {
569     unsigned long cw;
570     unsigned long sw;
571     unsigned long tag;
572     unsigned long ipoff;
573     unsigned long cssel;
574     unsigned long dataoff;
575     unsigned long datasel;
576     struct _libc_fpreg _st[8];
577     unsigned long status;
578 }

```

```

579     ;
580     typedef struct _libc_fpstate *fpregset_t;
581
582     typedef struct
583     {
584         gregset_t gregs;
585         fpregset_t fpregs;
586         unsigned long oldmask;
587         unsigned long cr2;
588     }
589     mcontext_t;
590
591     typedef struct ucontext
592     {
593         unsigned long uc_flags;
594         struct ucontext *uc_link;
595         stack_t uc_stack;
596         mcontext_t uc_mcontext;
597         sigset_t uc_sigmask;
598         struct _libc_fpstate __fpregs_mem;
599     }
600     ucontext_t;

```

### 1.3.19. unistd.h

```

601
602     typedef int intptr_t;

```

### 1.3.20. utmp.h

```

603
604     struct lastlog
605     {
606         time_t ll_time;
607         char ll_line[UT_LINESIZE];
608         char ll_host[UT_HOSTSIZE];
609     }
610     ;
611
612     struct utmp
613     {
614         short ut_type;
615         pid_t ut_pid;
616         char ut_line[UT_LINESIZE];
617         char ut_id[4];
618         char ut_user[UT_NAMESIZE];
619         char ut_host[UT_HOSTSIZE];
620         struct exit_status ut_exit;
621         long ut_session;
622         struct timeval ut_tv;
623         int32_t ut_addr_v6[4];
624         char __unused[20];

```



```
625 }
626 ;
```

### 1.3.21. utmpx.h

```
627
628 struct utmpx
629 {
630     short ut_type;
631     pid_t ut_pid;
632     char ut_line[UT_LINESIZE];
633     char ut_id[4];
634     char ut_user[UT_NAMESIZE];
635     char ut_host[UT_HOSTSIZE];
636     struct exit_status ut_exit;
637     long ut_session;
638     struct timeval ut_tv;
639     int32_t ut_addr_v6[4];
640     char __unused[20];
641 }
642 ;
```

## 1.4. Interfaces for libm

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

644 **Table 1-28. libm Definition**

Library:	libm
SONAME:	libm.so.6

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

ISO C (1999)  
 SUSv2  
 647 ISO POSIX (2003)

### 1.4.1. Math

#### 1.4.1.1. Interfaces for Math

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

651 **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) [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]
asinhhl(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]	significantd(GLIBC_2.0) [2]
atanhf(GLIBC_2.0) [1]	cos(GLIBC_2.0) [1]	fetestexcept(GLIBC_2.1) [1]	log10(GLIBC_2.0) [1]	significantdf(GLIBC_2.0) [2]
atanhl(GLIBC_2.0) [1]	cosf(GLIBC_2.0) [1]	feupdateenv(GLIBC_2.2) [1]	log10f(GLIBC_2.0) [1]	significantdl(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]
caacos(GLIBC_2.1) [1]	cpow(GLIBC_2.1) [1]	floorf(GLIBC_2.0) [1]	logl(GLIBC_2.0) [1]	sinf(GLIBC_2.0) [1]
caacosf(GLIBC_2.1) [1]	cpowf(GLIBC_2.1) [1]	floorl(GLIBC_2.0) [1]	lrint(GLIBC_2.1) [1]	sinh(GLIBC_2.0) [1]
caacosh(GLIBC_2.1) [1]	cpowl(GLIBC_2.1) [1]	fma(GLIBC_2.1) [1]	lrintf(GLIBC_2.1) [1]	sinhf(GLIBC_2.0) [1]
caacoshf(GLIBC_2.1) [1]	cproj(GLIBC_2.1) [1]	fmaf(GLIBC_2.1) [1]	lrintl(GLIBC_2.1) [1]	sinhl(GLIBC_2.0) [1]
caacoshl(GLIBC_2.1) [1]	cprojf(GLIBC_2.1) [1]	fmal(GLIBC_2.1) [1]	lround(GLIBC_2.1) [1]	sinl(GLIBC_2.0) [1]
caacosl(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]	
ceil(GLIBC_2.0) [1]	exp(GLIBC_2.0) [1]	jn(GLIBC_2.0) [1]	remquo(GLIBC_2.1) [1]	

653 *Referenced Specification(s)*

654 [1]. ISO POSIX (2003)

655 [2]. ISO C (1999)

656 [3]. SUSv2

657 An LSB conforming implementation shall provide the architecture specific data interfaces for Math specified in Table  
658 1-30, with the full functionality as described in the referenced underlying specification.

659 **Table 1-30. libm - Math Data Interfaces**

660	signgam(GLIBC_2. 0) [1]				
-----	----------------------------	--	--	--	--

661 *Referenced Specification(s)*

662 [1]. ISO POSIX (2003)

## 1.5. Interfaces for libpthread

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

664 **Table 1-31. libpthread Definition**

665	Library:	libpthread
	SONAME:	libpthread.so.0

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

667 Large File Support  
this specification  
ISO POSIX (2003)

### 1.5.1. Realtime Threads

#### 1.5.1.1. Interfaces for Realtime Threads

669 No external functions are defined for libpthread - Realtime Threads

### 1.5.2. Advanced Realtime Threads

#### 1.5.2.1. Interfaces for Advanced Realtime Threads

671 No external functions are defined for libpthread - Advanced Realtime Threads

## 1.5.3. Posix Threads

### 1.5.3.1. Interfaces for Posix Threads

672 An LSB conforming implementation shall provide the architecture specific functions for Posix Threads specified in  
 673 Table 1-32, with the full functionality as described in the referenced underlying specification.  
 674

675 **Table 1-32. libpthread - Posix Threads Function Interfaces**

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

676

677 *Referenced Specification(s)*

678 [1]. this specification

679 [2]. ISO POSIX (2003)

680 [3]. Large File Support

## 1.6. Interfaces for libgcc\_s

681 Table 1-33 defines the library name and shared object name for the libgcc\_s library

682 **Table 1-33. libgcc\_s Definition**

Library:	libgcc_s
SONAME:	libgcc_s.so.1

683

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

685 this specification

### 1.6.1. Unwind Library

#### 1.6.1.1. Interfaces for Unwind Library

687 An LSB conforming implementation shall provide the architecture specific functions for Unwind Library specified in

688 Table 1-34, with the full functionality as described in the referenced underlying specification.

689 **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]
<code>_Unwind_Find_FDE(GCC_3.0)</code> [1]	<code>_Unwind_GetGR(GCC_3.0)</code> [1]	<code>_Unwind_GetRegionStart(GCC_3.0)</code> [1]	<code>_Unwind_Resume(GCC_3.0)</code> [1]	
<code>_Unwind_ForcedUnwind(GCC_3.0)</code> [1]	<code>_Unwind_GetIP(GCC_3.0)</code> [1]	<code>_Unwind_GetTextRelBase(GCC_3.0)</code> [1]	<code>_Unwind_SetGR(GCC_3.0)</code> [1]	

690

691 *Referenced Specification(s)*

692 [1], this specification

## 1.7. Interface Definitions for `libgcc_s`

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

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

### `_Unwind_DeleteException`

#### Name

696 `_Unwind_DeleteException` — private C++ error handling method

#### Synopsis

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

#### Description

698 `_Unwind_DeleteException` deletes the given exception *object*. If a given runtime resumes normal execution  
 699 after catching a foreign exception, it will not know how to delete that exception. Such an exception shall be deleted by  
 700 calling `_Unwind_DeleteException`. This is a convenience function that calls the function pointed to by the  
 701 *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**

705 `_Unwind_ForcedUnwind` — private C++ error handling method

### **Synopsis**

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

### **Description**

708 `_Unwind_ForcedUnwind` raises an exception for forced unwinding, passing along the given exception *object*,  
709 which should have its *exception\_class* and *exception\_cleanup* fields set. The exception *object* has been allocated by  
710 the language-specific runtime, and has a language-specific format, except that it shall contain an `_Unwind_Exception`  
711 struct.

712 Forced unwinding is a single-phase process. *stop* and *stop\_parameter* control the termination of the unwind  
713 process instead of the usual personality routine query. *stop* is called for each unwind frame, with the parameters  
714 described for the usual personality routine below, plus an additional *stop\_parameter*.

### **Return Value**

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

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

720 `_URC_NO_REASON`

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

724 `_URC_END_OF_STACK`

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

728 `_URC_FATAL_PHASE2_ERROR`

729 The *stop* function may return this code for other fatal conditions like stack corruption.

## **`_Unwind_GetDataRelBase`**

### **Name**

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

### **Synopsis**

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

### **Description**

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

## **`_Unwind_GetGR`**

### **Name**

733 `_Unwind_GetGR` — private C++ error handling method

### **Synopsis**

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

### **Description**

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

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

## **`_Unwind_GetIP`**

### **Name**

739 `_Unwind_GetIP` — private C++ error handling method

### **Synopsis**

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

### **Description**

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

## **`_Unwind_GetLanguageSpecificData`**

### **Name**

742 `_Unwind_GetLanguageSpecificData` — private C++ error handling method

### **Synopsis**

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

### **Description**

745 `_Unwind_GetLanguageSpecificData` returns the address of the language specific data area for the current stack  
746 frame.

## **`_Unwind_GetRegionStart`**

### **Name**

747 `_Unwind_GetRegionStart` — private C++ error handling method

### **Synopsis**

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

### **Description**

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

## **`_Unwind_GetTextRelBase`**

### **Name**

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

### **Synopsis**

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

### **Description**

753 `_Unwind_GetTextRelBase` calls the abort method, then returns.

## **`_Unwind_RaiseException`**

### **Name**

754 `_Unwind_RaiseException` — private C++ error handling method

### **Synopsis**

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

### **Description**

756 `_Unwind_RaiseException` raises an exception, passing along the given exception *object*, which should have its  
 757 *exception\_class* and *exception\_cleanup* fields set. The exception object has been allocated by the  
 758 language-specific runtime, and has a language-specific format, exception that it shall contain an  
 759 `_Unwind_Exception`.

### **Return Value**

760 `_Unwind_RaiseException` does not return unless an error condition is found. If an error condition occurs, an  
 761 `_Unwind_Reason_Code` is returned:

762 `_URC_END_OF_STACK`

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

765 `_URC_FATAL_PHASE1_ERROR`

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

769 `_URC_FATAL_PHASE2_ERROR`

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

## **`_Unwind_Resume`**

### **Name**

772 `_Unwind_Resume` — private C++ error handling method

### **Synopsis**

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

### **Description**

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

## **`_Unwind_SetGR`**

### **Name**

776 `_Unwind_SetGR` — private C++ error handling method

### **Synopsis**

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

### **Description**

778 `_Unwind_SetGR` sets the *value* of the register *indexed* for the routine identified by the unwind *context*.

## **`_Unwind_SetIP`**

### **Name**

779 `_Unwind_SetIP` — private C++ error handling method

### **Synopsis**

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

### **Description**

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

## **1.8. Interfaces for libdl**

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

783 **Table 1-35. libdl Definition**

Library:	libdl
SONAME:	libdl.so.2

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

this specification

786 ISO POSIX (2003)

## 1.8.1. Dynamic Loader

### 787 1.8.1.1. Interfaces for Dynamic Loader

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

790 **Table 1-36. libdl - Dynamic Loader Function Interfaces**

dldladdr(GLIBC_2.0) [1]	dldlclose(GLIBC_2.0) [2]	dldlerror(GLIBC_2.0) [2]	dldlopen(GLIBC_2.1) [1]	dldlsym(GLIBC_2.0) [1]
----------------------------	-----------------------------	-----------------------------	----------------------------	---------------------------

792 *Referenced Specification(s)*

793 [1]. this specification

794 [2]. ISO POSIX (2003)

## 1.9. Interfaces for libcrypt

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

796 **Table 1-37. libcrypt Definition**

Library:	libcrypt
SONAME:	libcrypt.so.1

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

799 ISO POSIX (2003)

### 1.9.1. Encryption

#### 800 1.9.1.1. Interfaces for Encryption

801 An LSB conforming implementation shall provide the architecture specific functions for Encryption specified in Table  
802 1-38, with the full functionality as described in the referenced underlying specification.

803 **Table 1-38. libcrypt - Encryption Function Interfaces**

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

804	[1]	) [1]	[1]		
-----	-----	-------	-----	--	--

805 *Referenced Specification(s)*

806 [1]. ISO POSIX (2003)



## **II. Utility Libraries**

# Chapter 2. Libraries

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

## 2.1. Interfaces for libz

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

5 **Table 2-1. libz Definition**

Library:	libz
SONAME:	libz.so.1

### 2.1.1. Compression Library

#### 2.1.1.1. Interfaces for Compression Library

8 No external functions are defined for libz - Compression Library

## 2.2. Interfaces for libncurses

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

10 **Table 2-2. libncurses Definition**

Library:	libncurses
SONAME:	libncurses.so.5

### 2.2.1. Curses

#### 2.2.1.1. Interfaces for Curses

13 No external functions are defined for libncurses - Curses

## 2.3. Interfaces for libutil

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

15 **Table 2-3. libutil Definition**

Library:	libutil
SONAME:	libutil.so.1

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

## 2.3.1. Utility Functions

### 19 2.3.1.1. Interfaces for Utility Functions

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

22 **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]		

24 *Referenced Specification(s)*

25 [1]. this specification

# Appendix A. Alphabetical Listing of Interfaces

## A.1. libgcc\_s

- 1 The behaviour of the interfaces in this library is specified by the following Standards.  
2 this specification

3 **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]	

4

# **Linux Packaging Specification**

2

3 **Linux Packaging Specification**

# Table of Contents

<b>I. Package Format and Installation</b> .....	<b>50</b>
1. Software Installation .....	1
1.1. Package Dependencies.....	1
1.2. Package Architecture Considerations .....	1

# **I. Package Format and Installation**



# Chapter 1. Software Installation

## 1.1. Package Dependencies

- 1 The LSB runtime environment shall provide the following dependencies.
- 2 `lsb-core-ia32`
  - 3 This dependency is used to indicate that the application is dependent on features contained in the LSB-Core
  - 4 specification.
- 5 Other LSB modules may add additional dependencies; such dependencies shall have the format `lsb-module-ia32`.

## 1.2. Package Architecture Considerations

- 6 All packages must specify an architecture of `i486`. A LSB runtime environment must accept an architecture of `i486`
- 7 even if the native architecture is different.
- 8 The `archnum` value in the Lead Section shall be `0x0001`.

# **Free Documentation License**

2

3 **Free Documentation License**

# Table of Contents

<b>A. GNU Free Documentation License</b> .....	<b>1</b>
A.1. PREAMBLE.....	1
A.2. APPLICABILITY AND DEFINITIONS.....	1
A.3. VERBATIM COPYING.....	2
A.4. COPYING IN QUANTITY.....	2
A.5. MODIFICATIONS.....	3
A.6. COMBINING DOCUMENTS.....	4
A.7. COLLECTIONS OF DOCUMENTS.....	4
A.8. AGGREGATION WITH INDEPENDENT WORKS.....	4
A.9. TRANSLATION.....	5
A.10. TERMINATION.....	5
A.11. FUTURE REVISIONS OF THIS LICENSE.....	5
A.12. How to use this License for your documents.....	5

# Appendix A. GNU Free Documentation License

1 Version 1.1, March 2000

2 Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is  
3 permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## A.1. PREAMBLE

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

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

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

## A.2. APPLICABILITY AND DEFINITIONS

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

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

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

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

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

30 A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification  
31 is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic  
32 text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available  
33 drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats  
34 suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been

35 designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not  
36 "Transparent" is called "Opaque".

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

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

### **A.3. VERBATIM COPYING**

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

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

### **A.4. COPYING IN QUANTITY**

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

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

62 If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a  
63 machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a  
64 publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of  
65 added material, which the general network-using public has access to download anonymously at no charge using  
66 public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you  
67 begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the  
68 stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents  
69 or retailers) of that edition to the public.

70 It is requested, but not required, that you contact the authors of the Document well before redistributing any large  
71 number of copies, to give them a chance to provide you with an updated version of the Document.

## A.5. MODIFICATIONS

72 You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above,  
73 provided that you release the Modified Version under precisely this License, with the Modified Version filling the role  
74 of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of  
75 it. In addition, you must do these things in the Modified Version:

76 A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of  
77 previous versions (which should, if there were any, be listed in the History section of the Document). You may  
78 use the same title as a previous version if the original publisher of that version gives permission.

79 B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications  
80 in the Modified Version, together with at least five of the principal authors of the Document (all of its principal  
81 authors, if it has less than five).

82 C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

83 D. Preserve all the copyright notices of the Document.

84 E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

85 F. Include, immediately after the copyright notices, a license notice giving the public permission to use the  
86 Modified Version under the terms of this License, in the form shown in the Addendum below.

87 G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the  
88 Document's license notice.

89 H. Include an unaltered copy of this License.

90 I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new  
91 authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History"  
92 in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title  
93 Page, then add an item describing the Modified Version as stated in the previous sentence.

94 J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the  
95 Document, and likewise the network locations given in the Document for previous versions it was based on.  
96 These may be placed in the "History" section. You may omit a network location for a work that was published at  
97 least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

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

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

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

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

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

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

111 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  
112 Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of  
113 Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already  
114 includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are  
115 acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the  
116 previous publisher that added the old one.

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

## A.6. COMBINING DOCUMENTS

119 You may combine the Document with other documents released under this License, under the terms defined in section  
120 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the  
121 original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

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

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

## A.7. COLLECTIONS OF DOCUMENTS

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

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

## A.8. AGGREGATION WITH INDEPENDENT WORKS

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

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



## A.9. TRANSLATION

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

## A.10. TERMINATION

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

## A.11. FUTURE REVISIONS OF THIS LICENSE

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

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

## A.12. How to use this License for your documents

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

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

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

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