

Linux Standard Base Core Specification for AMD64 3.1

Linux Standard Base Core Specification for AMD64 3.1

Copyright © 2004, 2005 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.

PowerPC and PowerPC Architecture are trademarks of the IBM Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Contents

Foreword	vi
Introduction	vii
I Introductory Elements	8
1 Scope.....	9
1.1 General.....	9
1.2 Module Specific Scope.....	9
2 References	10
2.1 Normative References	10
2.2 Informative References/Bibliography	12
3 Requirements	15
3.1 Relevant Libraries	15
3.2 LSB Implementation Conformance	15
3.3 LSB Application Conformance.....	16
4 Definitions	18
5 Terminology	19
6 Documentation Conventions	21
II Executable and Linking Format (ELF)	22
7 Introduction.....	23
8 Low Level System Information.....	24
8.1 Machine Interface.....	24
8.2 Function Calling Sequence.....	25
8.3 Operating System Interface	26
8.4 Process Initialization.....	26
8.5 Coding Examples	27
8.6 C Stack Frame	27
8.7 Debug Information.....	27
9 Object Format	28
9.1 Introduction	28
9.2 ELF Header	28
9.3 Sections.....	28
9.4 Symbol Table	29
9.5 Relocation.....	29
10 Program Loading and Dynamic Linking	30
10.1 Introduction	30
10.2 Program Header.....	30
10.3 Program Loading	30
10.4 Dynamic Linking.....	30
III Base Libraries	32
11 Libraries	33
11.1 Program Interpreter/Dynamic Linker	33
11.2 Interfaces for libc	33
11.3 Data Definitions for libc	48
11.4 Interfaces for libm	73
11.5 Data Definitions for libm.....	78
11.6 Interfaces for libpthread.....	84
11.7 Data Definitions for libpthread	86
11.8 Interfaces for libgcc_s	91
11.9 Data Definitions for libgcc_s.....	92
11.10 Interface Definitions for libgcc_s.....	95

11.11 Interfaces for libdl	100
11.12 Data Definitions for libdl	101
11.13 Interfaces for libcrypt.....	101
IV Utility Libraries.....	102
12 Libraries	103
12.1 Interfaces for libz.....	103
12.2 Data Definitions for libz	103
12.3 Interfaces for libncurses.....	104
12.4 Data Definitions for libncurses.....	104
12.5 Interfaces for libutil.....	110
V Package Format and Installation	111
13 Software Installation	112
13.1 Package Dependencies	112
13.2 Package Architecture Considerations	112
A Alphabetical Listing of Interfaces.....	113
A.1 libgcc_s.....	113
A.2 libm.....	113
B GNU Free Documentation License (Informative)	114
B.1 PREAMBLE	114
B.2 APPLICABILITY AND DEFINITIONS.....	114
B.3 VERBATIM COPYING	115
B.4 COPYING IN QUANTITY	115
B.5 MODIFICATIONS	116
B.6 COMBINING DOCUMENTS.....	117
B.7 COLLECTIONS OF DOCUMENTS.....	118
B.8 AGGREGATION WITH INDEPENDENT WORKS.....	118
B.9 TRANSLATION	118
B.10 TERMINATION	118
B.11 FUTURE REVISIONS OF THIS LICENSE	119
B.12 How to use this License for your documents.....	119

List of Tables

2-1 Normative References	10
2-2 Other References	12
3-1 Standard Library Names	15
8-1 Non Conforming Instructions	24
9-1 ELF Special Sections	28
9-2 Additional Special Sections	29
11-1 libc Definition	33
11-2 libc - RPC Function Interfaces	33
11-3 libc - System Calls Function Interfaces	34
11-4 libc - Standard I/O Function Interfaces	36
11-5 libc - Standard I/O Data Interfaces	37
11-6 libc - Signal Handling Function Interfaces	38
11-7 libc - Signal Handling Data Interfaces	38
11-8 libc - Localization Functions Function Interfaces	38
11-9 libc - Localization Functions Data Interfaces	39
11-10 libc - Socket Interface Function Interfaces	39
11-11 libc - Wide Characters Function Interfaces	40
11-12 libc - String Functions Function Interfaces	41
11-13 libc - IPC Functions Function Interfaces	42
11-14 libc - Regular Expressions Function Interfaces	42
11-15 libc - Character Type Functions Function Interfaces	42
11-16 libc - Time Manipulation Function Interfaces	43
11-17 libc - Time Manipulation Data Interfaces	43
11-18 libc - Terminal Interface Functions Function Interfaces	44
11-19 libc - System Database Interface Function Interfaces	44
11-20 libc - Language Support Function Interfaces	45
11-21 libc - Large File Support Function Interfaces	45
11-22 libc - Standard Library Function Interfaces	45
11-23 libc - Standard Library Data Interfaces	48
11-24 libm Definition	73
11-25 libm - Math Function Interfaces	74
11-26 libm - Math Data Interfaces	78
11-27 libpthread Definition	84
11-28 libpthread - Realtime Threads Function Interfaces	84
11-29 libpthread - Posix Threads Function Interfaces	85
11-30 libpthread - Thread aware versions of libc interfaces Function Interfaces	86
11-31 libgcc_s Definition	91
11-32 libgcc_s - Unwind Library Function Interfaces	91
11-33 libdl Definition	100
11-34 libdl - Dynamic Loader Function Interfaces	100
11-35 libcrypt Definition	101
11-36 libcrypt - Encryption Function Interfaces	101
12-1 libz Definition	103
12-2 libncurses Definition	104
12-3 libutil Definition	110
12-4 libutil - Utility Functions Function Interfaces	110
A-1 libgcc_s Function Interfaces	113
A-2 libm Function Interfaces	113

Foreword

1 This is version 3.1 of the Linux Standard Base Core Specification for AMD64. This
2 specification is part of a family of specifications under the general title "Linux
3 Standard Base". Developers of applications or implementations interested in using
4 the LSB trademark should see the Free Standards Group Certification Policy for
5 details.

Introduction

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

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

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

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

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

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

I Introductory Elements

1 Scope

1.1 General

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

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

13 The LSB-generic document shall be used in conjunction with an architecture-specific
14 supplement. Whenever a section of the LSB-generic specification shall be
15 supplemented by architecture-specific information, the LSB-generic document
16 includes a reference to the architecture supplement. Architecture supplements may
17 also contain additional information that is not referenced in the LSB-generic
18 document.

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

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

1.2 Module Specific Scope

28 This is the AMD64 architecture specific Core module of the Linux Standards Base
29 (LSB). This module supplements the generic LSB Core module with those interfaces
30 that differ between architectures.

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

2 References

2.1 Normative References

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

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

10 **Table 2-1 Normative References**

Name	Title	URL
AMD64 Architecture Programmer's Manual, Volume 1	AMD64 Architecture Programmer's Manual, Volume 1: Application Programming 24592 3.08	http://www.amd.com/us-en/Processors/DevelopmentWithAMD/
AMD64 Architecture Programmer's Manual, Volume 2	AMD64 Architecture Programmer's Manual, Volume 2: System Programming 24593 3.08	http://www.amd.com/us-en/Processors/DevelopmentWithAMD/
AMD64 Architecture Programmer's Manual, Volume 3	AMD64 Architecture Programmer's Manual, Volume 3: General Purpose and System Instructions 24594 3.03	http://www.amd.com/us-en/Processors/DevelopmentWithAMD/
AMD64 Architecture Programmer's Manual, Volume 4	AMD64 Architecture Programmer's Manual, Volume 4: 128-bit Media Instructions 26568 3.04	http://www.amd.com/us-en/Processors/DevelopmentWithAMD/
AMD64 Architecture Programmer's Manual, Volume 5	AMD64 Architecture Programmer's Manual, Volume 5: 64-bit Media and x87 Floating-Point Instructions 26569 3.03	http://www.amd.com/us-en/Processors/DevelopmentWithAMD/
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEC 60559/IEEE 754 Floating Point	IEC 60559:1989 Binary floating-point arithmetic for microprocessor systems	http://www.ieee.org/
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages	

Name	Title	URL
	--C	
ISO POSIX (2003)	<p>ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions</p> <p>ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces</p> <p>ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities</p> <p>ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004</p>	http://www.unix.org/ version3/
Large File Support	Large File Support	http://www.UNIX-systems.org/ version2/whatsnew/ lfs20mar.html
SUSv2	CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)	http://www.opengroup.org/ publications/catalog/ un.htm
SUSv2 Commands and Utilities	The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/ publications/catalog/ un.htm
SVID Issue 3	American Telephone and Telegraph Company, System V Interface Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989.(ISBN 0201566524)	

Name	Title	URL
SVID Issue 4	System V Interface Definition, Fourth Edition	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
System V Application Binary Interface AMD64 Architecture Processor Supplement	System V Application Binary Interface AMD64 Architecture Processor Supplement, Draft Version 0.95	http://www.x86-64.org/documentation/abi-0.95.pdf
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	http://www.opengroup.org/publications/catalog/un.htm

11

2.2 Informative References/Bibliography

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

15 **Table 2-2 Other References**

Name	Title	URL
DWARF Debugging Information Format, Revision 2.0.0	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf
DWARF Debugging Information Format, Revision 3.0.0 (Draft)	DWARF Debugging Information Format, Revision 3.0.0 (Draft)	http://refspecs.freestandards.org/dwarf/
ISO/IEC TR14652	ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions	
ITU-T V.42	International Telecommunication Union Recommendation V.42 (2002): Error-correcting procedures for DCEs using asynchronous-to-synchro	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42

Name	Title	URL
	nous conversionITUV	
Li18nux Globalization Specification	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt
PAM	Open Software Foundation, Request For Comments: 86.0 , October 1995, V. Samar & R.Schemers (SunSoft)	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 1831/1832 RPC & XDR	IETF RFC 1831 & 1832	http://www.ietf.org/
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1950: ZLIB Compressed Data Format Specification	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821:Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822:Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 791:Internet Protocol	IETF RFC 791: Internet Protocol Specification	http://www.ietf.org/rfc/rfc791.txt
RPM Package Format	RPM Package Format V3.0	http://www.rpm.org/max-rpm/s1-rpm-file-format-rpm-file-format.html

2 References

Name	Title	URL
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

16

3 Requirements

3.1 Relevant Libraries

1 The libraries listed in Table 3-1 shall be available on x86-64 Linux Standard Base
2 systems, with the specified runtime names. These names override or supplement the
3 names specified in the generic LSB specification. The specified program interpreter,
4 referred to as proginterp in this table, shall be used to load the shared libraries
5 specified by DT_NEEDED entries at run time.

6 **Table 3-1 Standard Library Names**

Library	Runtime Name
libm	libm.so.6
libdl	libdl.so.2
libcrypt	libcrypt.so.1
libz	libz.so.1
libncurses	libncurses.so.5
libutil	libutil.so.1
libc	libc.so.6
libpthread	libpthread.so.0
proginterp	/lib64/ld-lsb-x86-64.so.3
libgcc_s	libgcc_s.so.1

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

3.2 LSB Implementation Conformance

10 A conforming implementation is necessarily architecture specific, and must provide
11 the interfaces specified by both the generic LSB Core specification and its relevant
12 architecture specific supplement.

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

15 A conforming implementation shall satisfy the following requirements:

- 16 • A processor architecture represents a family of related processors which may not
17 have identical feature sets. The architecture specific supplement to this
18 specification for a given target processor architecture describes a minimum
19 acceptable processor. The implementation shall provide all features of this
20 processor, whether in hardware or through emulation transparent to the
21 application.
- 22 • The implementation shall be capable of executing compiled applications having
23 the format and using the system interfaces described in this document.
- 24 • The implementation shall provide libraries containing the interfaces specified by
25 this document, and shall provide a dynamic linking mechanism that allows these

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

3.3 LSB Application Conformance

51 A conforming application is necessarily architecture specific, and must conform to
52 both the generic LSB Core specification and its relevant architecture specific
53 supplement.

54 A conforming application shall satisfy the following requirements:

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

69 • The use of that interface or data format, as well as its source, shall be identified
70 in the documentation of the application.

71 • It shall not use any values for a named interface that are reserved for vendor
72 extensions.

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

4 Definitions

1	For the purposes of this document, the following definitions, as specified in the
2	<i>ISO/IEC Directives, Part 2, 2001, 4th Edition</i> , 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
15	required that...be not; is not to be
16	should
17	it is recommended that; ought to
18	should not
19	it is not recommended that; ought not to

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
4 of the interface that are platform specific. The archLSB is complementary to the
5 gLSB.

6 Binary Standard

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

9 gLSB

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

12 implementation-defined

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

20 Shell Script

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

23 Source Standard

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

26 undefined

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

33 unspecified

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

5 Terminology

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

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
13 in these tables has the following 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
20 this table.

21 For example,

22 `forkpty(GLIBC_2.0) [SUSv3]`

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

25 **Note:** Symbol versions are defined in the architecture specific supplements only.

II Executable and Linking Format (ELF)

7 Introduction

1 Executable and Linking Format (ELF) defines the object format for compiled
2 applications. This specification supplements the information found in System V ABI
3 Update and System V Application Binary Interface AMD64 Architecture Processor
4 Supplement, and is intended to document additions made since the publication of
5 that document.

8 Low Level System Information

8.1 Machine Interface

8.1.1 Processor Architecture

The AMD64 Architecture is specified by the following documents

- AMD64 Architecture Programmer's Manual, Volume 1
- AMD64 Architecture Programmer's Manual, Volume 2
- AMD64 Architecture Programmer's Manual, Volume 3
- AMD64 Architecture Programmer's Manual, Volume 4
- AMD64 Architecture Programmer's Manual, Volume 5
- System V Application Binary Interface AMD64 Architecture Processor Supplement

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

Applications conforming to this specification should attempt to execute in a diminished capacity if a required instruction set feature is not present. In particular, applications should not rely on the availability of the 3DNow!™ technology. In addition, a conforming application shall not use any instruction from Table 8-1.

Note: Although this specification carries the attribution "AMD64", it is intended to apply to the entire `x86_64` set of processors, including those based on Intel® Extended Memory 64 Technology (EM64T). However, this specification defers to the AMD architecture specified above.

Table 8-1 Non Conforming Instructions

LAHF	SAHF
SYSCALL	SYSRET
SYSENTER	SYSEXIT
CMPXCHG16B	FFXSR

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

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

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

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

8.1.2 Data Representation

8.1.2.1 Introduction

32
33
34
35

LSB-conforming applications shall use the data representation as defined in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

36
37
38

Note: The System V Application Binary Interface AMD64 Architecture Processor Supplement specification is itself layered on top of the System V Application Binary Interface - Intel386™ Architecture Processor Supplement.

8.1.2.2 Byte Ordering

39
40
41

LSB-conforming applications shall use the byte ordering defined in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.1.2.3 Fundamental Types

42
43
44
45

LSB-conforming applications shall use only the fundamental types described in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.1.2.4 Aggregates and Unions

46
47
48
49

LSB-conforming applications shall use alignment for aggregates and unions as described in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.1.2.5 Bit Fields

50
51
52
53

LSB-conforming applications utilizing bit-fields shall follow the requirements of Section 3.1.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.2 Function Calling Sequence

8.2.1 Introduction

54
55
56

LSB-conforming applications shall use only the following features of the function calling sequence as defined in Section 3.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.2.2 Registers

57
58
59

LSB-conforming applications shall use only the registers described in Section 3.2.1 (Registers and the Stack Frame) of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.2.3 Floating Point Registers

60
61
62

LSB-conforming applications shall use only the floating point registers described in Section 3.2.1 (Registers and the Stack Frame) of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.2.4 Stack Frame

63 LSB-conforming applications shall use stack frames as described in Section 3.2.2 of
64 the System V Application Binary Interface AMD64 Architecture Processor
65 Supplement.

8.2.5 Arguments

66 LSB-conforming applications shall pass parameters to functions as described in
67 Section 3.2.3 of the System V Application Binary Interface AMD64 Architecture
68 Processor Supplement.

8.2.6 Return Values

69 Values are returned from functions as described in Section 3.3.2 of the System V
70 Application Binary Interface AMD64 Architecture Processor Supplement.

8.3 Operating System Interface

71 LSB-conforming applications shall use only the following features of the Operating
72 System Interfaces as defined in Section 3.3 of the System V Application Binary
73 Interface AMD64 Architecture Processor Supplement.

8.3.1 Exception Interface

74 Synchronous and floating point or coprocessor exceptions shall behave as described
75 in Section 3.3.1 of the System V Application Binary Interface AMD64 Architecture
76 Processor Supplement.

8.3.2 Virtual Address Space

77 LSB-Conforming applications shall use only the virtual address space described in
78 Section 3.3.2 and 3.3.4 of the System V Application Binary Interface AMD64
79 Architecture Processor Supplement. Virtual memory page sizes shall be subject to
80 the limitations described in Section 3.3.3 of the System V Application Binary
81 Interface AMD64 Architecture Processor Supplement.

8.4 Process Initialization

82 LSB-conforming applications shall use only the following features of the Process
83 Initialization as defined in Section 3.4 of the System V Application Binary Interface
84 AMD64 Architecture Processor Supplement.

8.4.1 Special Registers

85 During process initialization, the special registers shall be initialized as described in
86 Section 3.4.1 of the System V Application Binary Interface AMD64 Architecture
87 Processor Supplement.

8.4.2 Process Stack (on entry)

88 The process stack shall be initialized as described in Section 3.4.1 of the System V
89 Application Binary Interface AMD64 Architecture Processor Supplement.

8.4.3 Auxiliary Vector

90 The auxiliary vector shall be initialized as described in Section 3.4.2 of the System V
91 Application Binary Interface AMD64 Architecture Processor Supplement.

8.5 Coding Examples

92 LSB-conforming applications may use the coding examples given in Section 3.5 of
 93 the System V Application Binary Interface AMD64 Architecture Processor
 94 Supplement to guide implementation of fundamental operations in the following
 95 areas.

8.5.1 Code Model Overview/Architecture Constraints

96 Section 3.5.1 of the System V Application Binary Interface AMD64 Architecture
 97 Processor Supplement describes a number of code models. LSB-Conforming
 98 applications may use any of these models except the Kernel and Large code models.

8.5.2 Position-Independent Function Prologue

99 LSB-conforming applications may follow the position-independent function
 100 prologue example in Section 3.5.3 of the System V Application Binary Interface
 101 AMD64 Architecture Processor Supplement.

8.5.3 Data Objects

102 LSB-conforming applications may follow the data objects examples in Section 3.5.4
 103 of the System V Application Binary Interface AMD64 Architecture Processor
 104 Supplement.

8.5.4 Function Calls

105 LSB-conforming applications may follow the function call examples in Section 3.5.5
 106 of the System V Application Binary Interface AMD64 Architecture Processor
 107 Supplement. See Chapter 3 of System V Application Binary Interface AMD64
 108 Architecture Processor Supplement.

8.5.5 Branching

109 LSB-conforming applications may follow the branching examples in Section 3.5.6 of
 110 the System V Application Binary Interface AMD64 Architecture Processor
 111 Supplement.

8.6 C Stack Frame

8.6.1 Variable Argument List

112 LSB-Conforming applications shall only use variable arguments to functions in the
 113 manner described in Section 3.5.7 of the System V Application Binary Interface
 114 AMD64 Architecture Processor Supplement.

8.7 Debug Information

115 LSB-Conforming applications may include DWARF debugging information. The
 116 DWARF Release Number and Register Number Mapping shall be as described in
 117 Section 3.6 of the System V Application Binary Interface AMD64 Architecture
 118 Processor Supplement.

9 Object Format

9.1 Introduction

1 LSB-conforming implementations shall support the Executable and Linking Format
2 (ELF) object file , as defined by the System V ABI , System V ABI Update , System V
3 Application Binary Interface AMD64 Architecture Processor Supplement and as
4 supplemented by the generic LSB specification and This Specification.

9.2 ELF Header

9.2.1 Machine Information

5 LSB-conforming applications shall identify the Machine Information as defined in
6 Section 4.1.1 of the System V Application Binary Interface AMD64 Architecture
7 Processor Supplement.

9.3 Sections

9.3.1 Introduction

8 In addition to the requirements for ELF sections described in the generic LSB Core
9 specification, conforming implementations shall support architecture specific
10 sections as described below.

11 **Note:** The System V Application Binary Interface AMD64 Architecture Processor
12 Supplement specifies some architecture specific section flags and section types that are
13 not required by LSB-conforming systems.

9.3.2 Special Sections

14 The following architecture-specific sections are defined in the System V Application
15 Binary Interface AMD64 Architecture Processor Supplement.

16 **Table 9-1 ELF Special Sections**

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

17
18 .got
19 This section holds the global offset table

20 .plt
21 This section holds the procedure linkage table.

22 **Note:** Since LSB-conforming implementations are not required to support the large code
23 model, it is not necessary for them to provide support for the additional special sections
24 for the large code model described in the System V Application Binary Interface AMD64
25 Architecture Processor Supplement.

26 Also, the System V Application Binary Interface AMD64 Architecture Processor
 27 Supplement specifies a section `.eh_frame`, with a type of `SHT_AMD64_UNWIND`. This
 28 section is described in the generic LSB-Core specification, but with type `SHT_PROGBITS`.
 29 This specification does not require support for the `SHT_AMD64_UNWIND` section type.

9.3.3 Additional Special Sections

30 The following additional sections are defined here.

31 **Table 9-2 Additional Special Sections**

Name	Type	Attributes
<code>.rela.dyn</code>	<code>SHT_RELA</code>	<code>SHF_ALLOC</code>
<code>.rela.plt</code>	<code>SHT_RELA</code>	<code>SHF_ALLOC</code>

32
 33 `.rela.dyn`

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

36 `.rela.plt`

37 This section holds RELA type relocation information for the PLT section of a
 38 shared library or dynamically linked application

9.4 Symbol Table

39 LSB-conforming applications shall use Symbol Tables as defined in Section 4.3 of the
 40 System V Application Binary Interface AMD64 Architecture Processor Supplement.

9.5 Relocation

41 LSB-conforming implementation shall support the required relocation types defined
 42 in Section 4.4.1 of the System V Application Binary Interface AMD64 Architecture
 43 Processor Supplement.

44 **Note:** Since LSB-conforming implementations are not required to support the large code
 45 model, it is not necessary for them to provide support for the additional relocation types
 46 for the large code model described in the System V Application Binary Interface AMD64
 47 Architecture Processor Supplement.

10 Program Loading and Dynamic Linking

10.1 Introduction

1 LSB-conforming implementations shall support the object file information and
2 system actions that create running programs as specified in the System V ABI ,
3 System V ABI Update , System V Application Binary Interface AMD64 Architecture
4 Processor Supplement and as supplemented by the generic LSB specification and
5 This Specification.

10.2 Program Header

6 LSB-conforming implementations are not required to support the additional types
7 and flags for this architecture as defined in Section 5.1 of the System V Application
8 Binary Interface AMD64 Architecture Processor Supplement.

9 **Note:** The System V Application Binary Interface AMD64 Architecture Processor
10 Supplement specification is itself layered on top of the System V Application Binary
11 Interface - Intel386™ Architecture Processor Supplement. As such, the requirements of
12 that specification are still requirements of this specification.

10.3 Program Loading

13 LSB-conforming implementations shall map file pages to virtual memory pages as
14 described in Section 5.1 of the System V Application Binary Interface AMD64
15 Architecture Processor Supplement.

10.4 Dynamic Linking

10.4.1 Introduction

16 LSB-conforming implementations shall provide dynamic linking as specified in
17 Section 5.2 of the System V Application Binary Interface AMD64 Architecture
18 Processor Supplement, except as described in the following sections.

19 **Note:** Since LSB-conforming implementations are not required to support the large
20 model, support for dynamic linking of large model code is not required.

10.4.2 Dynamic Section

21 Dynamic section entries give information to the dynamic linker. The following
22 dynamic entry types shall be supported:

23 DT_JMPREL

24 This entry is associated with a table of relocation entries for the procedure
25 linkage table. This entry is mandatory both for executable and shared object
26 files

27 DT_PLTGOT

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

30 DT_RELACOUNT

31 The number of relative relocations in .rela.dyn

10.4.3 Global Offset Table

32 LSB-conforming implementations shall support a Global Offset Table as described in
33 Section 5.2 of the System V Application Binary Interface AMD64 Architecture
34 Processor Supplement.

10.4.4 Function Addresses

35 Function addresses shall behave as described in Section 5.2 of the System V
36 Application Binary Interface AMD64 Architecture Processor Supplement.

10.4.5 Procedure Linkage Table

37 LSB-conforming implementations shall support a Procedure Linkage Table as
38 described in Section 5.2 of the System V Application Binary Interface AMD64
39 Architecture Processor Supplement.

10.4.6 Initialization and Termination Functions

40 LSB-conforming implementations shall support initialization and termination
41 functions as specified in Section 5.2.2 of the System V Application Binary Interface
42 AMD64 Architecture Processor Supplement.

III Base Libraries

11 Libraries

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

4 Interfaces that are unique to the AMD64 platform are defined here. This section
5 should be used in conjunction with the corresponding section in the Linux Standard
6 Base Specification.

11.1 Program Interpreter/Dynamic Linker

7 The Program Interpreter shall be `/lib64/ld-1sb-x86-64.so.3`.

11.2 Interfaces for libc

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

9 **Table 11-1 libc Definition**

Library:	libc
SONAME:	libc.so.6

10
11 The behavior of the interfaces in this library is specified by the following specifica-
12 tions:

[LFS] Large File Support
[LSB] This Specification
[SUSv2] SUSv2
[SUSv3] ISO POSIX (2003)
[SVID.3] SVID Issue 3
13 [SVID.4] SVID Issue 4

11.2.1 RPC

11.2.1.1 Interfaces for RPC

14
15 An LSB conforming implementation shall provide the architecture specific functions
16 for RPC specified in Table 11-2, with the full mandatory functionality as described in
17 the referenced underlying specification.

18 **Table 11-2 libc - RPC Function Interfaces**

authnone_create(GLIBC_2.2.5) [SVID.4]	clnt_create(GLIBC _2.2.5) [SVID.4]	clnt_pcreateerror(GLIBC_2.2.5) [SVID.4]	clnt_perrno(GLIB C_2.2.5) [SVID.4]
clnt_perror(GLIB C_2.2.5) [SVID.4]	clnt_screateerror (GLIBC_2.2.5) [SVID.4]	clnt_sperno(GLI BC_2.2.5) [SVID.4]	clnt_sperror(GLIB C_2.2.5) [SVID.4]
key_decryptsessio n(GLIBC_2.2.5) [SVID.3]	pmap_getport(GL IBC_2.2.5) [LSB]	pmap_set(GLIBC_ 2.2.5) [LSB]	pmap_unset(GLIB C_2.2.5) [LSB]
svc_getreqset(GLI	svc_register(GLIB	svc_run(GLIBC_2.	svc_sendreply(GL

BC_2.2.5) [SVID.3]	C_2.2.5) [LSB]	2.5) [LSB]	IBC_2.2.5) [LSB]
svcerr_auth(GLIBC_2.2.5) [SVID.3]	svcerr_decode(GLIBC_2.2.5) [SVID.3]	svcerr_noproc(GLIBC_2.2.5) [SVID.3]	svcerr_noprogram(GLIBC_2.2.5) [SVID.3]
svcerr_progvers(GLIBC_2.2.5) [SVID.3]	svcerr_systemerr(GLIBC_2.2.5) [SVID.3]	svcerr_weakauth(GLIBC_2.2.5) [SVID.3]	svctcp_create(GLIBC_2.2.5) [LSB]
svcdup_create(GLIBC_2.2.5) [LSB]	xdr_accepted_reply(GLIBC_2.2.5) [SVID.3]	xdr_array(GLIBC_2.2.5) [SVID.3]	xdr_bool(GLIBC_2.2.5) [SVID.3]
xdr_bytes(GLIBC_2.2.5) [SVID.3]	xdr_callhdr(GLIBC_2.2.5) [SVID.3]	xdr_callmsg(GLIBC_2.2.5) [SVID.3]	xdr_char(GLIBC_2.2.5) [SVID.3]
xdr_double(GLIBC_2.2.5) [SVID.3]	xdr_enum(GLIBC_2.2.5) [SVID.3]	xdr_float(GLIBC_2.2.5) [SVID.3]	xdr_free(GLIBC_2.2.5) [SVID.3]
xdr_int(GLIBC_2.2.5) [SVID.3]	xdr_long(GLIBC_2.2.5) [SVID.3]	xdr_opaque(GLIBC_2.2.5) [SVID.3]	xdr_opaque_auth(GLIBC_2.2.5) [SVID.3]
xdr_pointer(GLIBC_2.2.5) [SVID.3]	xdr_reference(GLIBC_2.2.5) [SVID.3]	xdr_rejected_reply(GLIBC_2.2.5) [SVID.3]	xdr_replymsg(GLIBC_2.2.5) [SVID.3]
xdr_short(GLIBC_2.2.5) [SVID.3]	xdr_string(GLIBC_2.2.5) [SVID.3]	xdr_u_char(GLIBC_2.2.5) [SVID.3]	xdr_u_int(GLIBC_2.2.5) [LSB]
xdr_u_long(GLIBC_2.2.5) [SVID.3]	xdr_u_short(GLIBC_2.2.5) [SVID.3]	xdr_union(GLIBC_2.2.5) [SVID.3]	xdr_vector(GLIBC_2.2.5) [SVID.3]
xdr_void(GLIBC_2.2.5) [SVID.3]	xdr_wrapstring(GLIBC_2.2.5) [SVID.3]	xdrmem_create(GLIBC_2.2.5) [SVID.3]	xdrrec_create(GLIBC_2.2.5) [SVID.3]
xdrrec_eof(GLIBC_2.2.5) [SVID.3]			

19

11.2.2 System Calls

20

11.2.2.1 Interfaces for System Calls

21

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

22

23

Table 11-3 libc - System Calls Function Interfaces

__fxstat(GLIBC_2.2.5) [LSB]	__getpgid(GLIBC_2.2.5) [LSB]	__lxstat(GLIBC_2.2.5) [LSB]	__xmknod(GLIBC_2.2.5) [LSB]
__xstat(GLIBC_2.2.5) [LSB]	access(GLIBC_2.2.5) [SUSv3]	acct(GLIBC_2.2.5) [LSB]	alarm(GLIBC_2.2.5) [SUSv3]
brk(GLIBC_2.2.5) [SUSv2]	chdir(GLIBC_2.2.5) [SUSv3]	chmod(GLIBC_2.2.5) [SUSv3]	chown(GLIBC_2.2.5) [SUSv3]

24

chroot(GLIBC_2.2.5) [SUSv2]	clock(GLIBC_2.2.5) [SUSv3]	close(GLIBC_2.2.5) [SUSv3]	closedir(GLIBC_2.2.5) [SUSv3]
creat(GLIBC_2.2.5) [SUSv3]	dup(GLIBC_2.2.5) [SUSv3]	dup2(GLIBC_2.2.5) [SUSv3]	execl(GLIBC_2.2.5) [SUSv3]
execle(GLIBC_2.2.5) [SUSv3]	execlp(GLIBC_2.2.5) [SUSv3]	execv(GLIBC_2.2.5) [SUSv3]	execve(GLIBC_2.2.5) [SUSv3]
execvp(GLIBC_2.2.5) [SUSv3]	exit(GLIBC_2.2.5) [SUSv3]	fchdir(GLIBC_2.2.5) [SUSv3]	fchmod(GLIBC_2.2.5) [SUSv3]
fchown(GLIBC_2.2.5) [SUSv3]	fcntl(GLIBC_2.2.5) [LSB]	fdatasync(GLIBC_2.2.5) [SUSv3]	flock(GLIBC_2.2.5) [LSB]
fork(GLIBC_2.2.5) [SUSv3]	fstatvfs(GLIBC_2.2.5) [SUSv3]	fsync(GLIBC_2.2.5) [SUSv3]	ftime(GLIBC_2.2.5) [SUSv3]
fruncate(GLIBC_2.2.5) [SUSv3]	getcontext(GLIBC_2.2.5) [SUSv3]	getegid(GLIBC_2.2.5) [SUSv3]	geteuid(GLIBC_2.2.5) [SUSv3]
getgid(GLIBC_2.2.5) [SUSv3]	getgroups(GLIBC_2.2.5) [SUSv3]	getitimer(GLIBC_2.2.5) [SUSv3]	getloadavg(GLIBC_2.2.5) [LSB]
getpagesize(GLIBC_2.2.5) [SUSv2]	getpgid(GLIBC_2.2.5) [SUSv3]	getpgrp(GLIBC_2.2.5) [SUSv3]	getpid(GLIBC_2.2.5) [SUSv3]
getppid(GLIBC_2.2.5) [SUSv3]	getpriority(GLIBC_2.2.5) [SUSv3]	getrlimit(GLIBC_2.2.5) [SUSv3]	getrusage(GLIBC_2.2.5) [SUSv3]
getsid(GLIBC_2.2.5) [SUSv3]	getuid(GLIBC_2.2.5) [SUSv3]	getwd(GLIBC_2.2.5) [SUSv3]	initgroups(GLIBC_2.2.5) [LSB]
ioctl(GLIBC_2.2.5) [LSB]	kill(GLIBC_2.2.5) [LSB]	killpg(GLIBC_2.2.5) [SUSv3]	lchown(GLIBC_2.2.5) [SUSv3]
link(GLIBC_2.2.5) [LSB]	lockf(GLIBC_2.2.5) [SUSv3]	lseek(GLIBC_2.2.5) [SUSv3]	mkdir(GLIBC_2.2.5) [SUSv3]
mkfifo(GLIBC_2.2.5) [SUSv3]	mlock(GLIBC_2.2.5) [SUSv3]	mlockall(GLIBC_2.2.5) [SUSv3]	mmap(GLIBC_2.2.5) [SUSv3]
mprotect(GLIBC_2.2.5) [SUSv3]	msync(GLIBC_2.2.5) [SUSv3]	munlock(GLIBC_2.2.5) [SUSv3]	munlockall(GLIBC_2.2.5) [SUSv3]
munmap(GLIBC_2.2.5) [SUSv3]	nanosleep(GLIBC_2.2.5) [SUSv3]	nice(GLIBC_2.2.5) [SUSv3]	open(GLIBC_2.2.5) [SUSv3]
opendir(GLIBC_2.2.5) [SUSv3]	pathconf(GLIBC_2.2.5) [SUSv3]	pause(GLIBC_2.2.5) [SUSv3]	pipe(GLIBC_2.2.5) [SUSv3]
poll(GLIBC_2.2.5) [SUSv3]	read(GLIBC_2.2.5) [SUSv3]	readdir(GLIBC_2.2.5) [SUSv3]	readdir_r(GLIBC_2.2.5) [SUSv3]
readlink(GLIBC_2.2.5) [SUSv3]	readv(GLIBC_2.2.5) [SUSv3]	rename(GLIBC_2.2.5) [SUSv3]	rmdir(GLIBC_2.2.5) [SUSv3]
sbrk(GLIBC_2.2.5) [SUSv2]	sched_get_priority_max(GLIBC_2.2.5) [SUSv3]	sched_get_priority_min(GLIBC_2.2.5) [SUSv3]	sched_getparam(GLIBC_2.2.5) [SUSv3]

sched_getscheduler(GLIBC_2.2.5) [SUSv3]	sched_rr_get_interval(GLIBC_2.2.5) [SUSv3]	sched_setparam(GLIBC_2.2.5) [SUSv3]	sched_setscheduler(GLIBC_2.2.5) [SUSv3]
sched_yield(GLIBC_2.2.5) [SUSv3]	select(GLIBC_2.2.5) [SUSv3]	setcontext(GLIBC_2.2.5) [SUSv3]	setegid(GLIBC_2.2.5) [SUSv3]
seteuid(GLIBC_2.2.5) [SUSv3]	setgid(GLIBC_2.2.5) [SUSv3]	setitimer(GLIBC_2.2.5) [SUSv3]	setpgid(GLIBC_2.2.5) [SUSv3]
setpgrp(GLIBC_2.2.5) [SUSv3]	setpriority(GLIBC_2.2.5) [SUSv3]	setregid(GLIBC_2.2.5) [SUSv3]	setreuid(GLIBC_2.2.5) [SUSv3]
setrlimit(GLIBC_2.2.5) [SUSv3]	setrlimit64(GLIBC_2.2.5) [LFS]	setsid(GLIBC_2.2.5) [SUSv3]	setuid(GLIBC_2.2.5) [SUSv3]
sleep(GLIBC_2.2.5) [SUSv3]	statvfs(GLIBC_2.2.5) [SUSv3]	stime(GLIBC_2.2.5) [LSB]	symlink(GLIBC_2.2.5) [SUSv3]
sync(GLIBC_2.2.5) [SUSv3]	sysconf(GLIBC_2.2.5) [SUSv3]	time(GLIBC_2.2.5) [SUSv3]	times(GLIBC_2.2.5) [SUSv3]
truncate(GLIBC_2.2.5) [SUSv3]	ulimit(GLIBC_2.2.5) [SUSv3]	umask(GLIBC_2.2.5) [SUSv3]	uname(GLIBC_2.2.5) [SUSv3]
unlink(GLIBC_2.2.5) [LSB]	utime(GLIBC_2.2.5) [SUSv3]	utimes(GLIBC_2.2.5) [SUSv3]	vfork(GLIBC_2.2.5) [SUSv3]
wait(GLIBC_2.2.5) [SUSv3]	wait4(GLIBC_2.2.5) [LSB]	waitpid(GLIBC_2.2.5) [LSB]	write(GLIBC_2.2.5) [SUSv3]
writev(GLIBC_2.2.5) [SUSv3]			

25

11.2.3 Standard I/O

26

11.2.3.1 Interfaces for Standard I/O

27

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

28

29

30

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

_IO_feof(GLIBC_2.2.5) [LSB]	_IO_getc(GLIBC_2.2.5) [LSB]	_IO_putc(GLIBC_2.2.5) [LSB]	_IO_puts(GLIBC_2.2.5) [LSB]
asprintf(GLIBC_2.2.5) [LSB]	clearerr(GLIBC_2.2.5) [SUSv3]	ctermid(GLIBC_2.2.5) [SUSv3]	fclose(GLIBC_2.2.5) [SUSv3]
fdopen(GLIBC_2.2.5) [SUSv3]	feof(GLIBC_2.2.5) [SUSv3]	ferror(GLIBC_2.2.5) [SUSv3]	fflush(GLIBC_2.2.5) [SUSv3]
fflush_unlocked(GLIBC_2.2.5) [LSB]	fgetc(GLIBC_2.2.5) [SUSv3]	fgetpos(GLIBC_2.2.5) [SUSv3]	fgets(GLIBC_2.2.5) [SUSv3]
fgetwc_unlocked(GLIBC_2.2.5)	fileno(GLIBC_2.2.5) [SUSv3]	flockfile(GLIBC_2.2.5) [SUSv3]	fopen(GLIBC_2.2.5) [SUSv3]

[LSB]			
fprintf(GLIBC_2.2.5) [SUSv3]	fputc(GLIBC_2.2.5) [SUSv3]	fputs(GLIBC_2.2.5) [SUSv3]	fread(GLIBC_2.2.5) [SUSv3]
freopen(GLIBC_2.2.5) [SUSv3]	fscanf(GLIBC_2.2.5) [LSB]	fseek(GLIBC_2.2.5) [SUSv3]	fseeko(GLIBC_2.2.5) [SUSv3]
fsetpos(GLIBC_2.2.5) [SUSv3]	ftell(GLIBC_2.2.5) [SUSv3]	ftello(GLIBC_2.2.5) [SUSv3]	fwrite(GLIBC_2.2.5) [SUSv3]
getc(GLIBC_2.2.5) [SUSv3]	getc_unlocked(GLIBC_2.2.5) [SUSv3]	getchar(GLIBC_2.2.5) [SUSv3]	getchar_unlocked(GLIBC_2.2.5) [SUSv3]
getw(GLIBC_2.2.5) [SUSv2]	pclose(GLIBC_2.2.5) [SUSv3]	popen(GLIBC_2.2.5) [SUSv3]	printf(GLIBC_2.2.5) [SUSv3]
putc(GLIBC_2.2.5) [SUSv3]	putc_unlocked(GLIBC_2.2.5) [SUSv3]	putchar(GLIBC_2.2.5) [SUSv3]	putchar_unlocked(GLIBC_2.2.5) [SUSv3]
puts(GLIBC_2.2.5) [SUSv3]	putw(GLIBC_2.2.5) [SUSv2]	remove(GLIBC_2.2.5) [SUSv3]	rewind(GLIBC_2.2.5) [SUSv3]
rewinddir(GLIBC_2.2.5) [SUSv3]	scanf(GLIBC_2.2.5) [LSB]	seekdir(GLIBC_2.2.5) [SUSv3]	setbuf(GLIBC_2.2.5) [SUSv3]
setbuffer(GLIBC_2.2.5) [LSB]	setvbuf(GLIBC_2.2.5) [SUSv3]	snprintf(GLIBC_2.2.5) [SUSv3]	sprintf(GLIBC_2.2.5) [SUSv3]
sscanf(GLIBC_2.2.5) [LSB]	telldir(GLIBC_2.2.5) [SUSv3]	tempnam(GLIBC_2.2.5) [SUSv3]	ungetc(GLIBC_2.2.5) [SUSv3]
vasprintf(GLIBC_2.2.5) [LSB]	vdprintf(GLIBC_2.2.5) [LSB]	vfprintf(GLIBC_2.2.5) [SUSv3]	vprintf(GLIBC_2.2.5) [SUSv3]
vsnprintf(GLIBC_2.2.5) [SUSv3]	vsprintf(GLIBC_2.2.5) [SUSv3]		

31

32

33

34

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

35

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

stderr(GLIBC_2.2.5) [SUSv3]	stdin(GLIBC_2.2.5) [SUSv3]	stdout(GLIBC_2.2.5) [SUSv3]	
-----------------------------	----------------------------	-----------------------------	--

36

11.2.4 Signal Handling

37

11.2.4.1 Interfaces for Signal Handling

38

39

40

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

41

Table 11-6 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.2.5) [LSB]	__libc_current_sigrtmin(GLIBC_2.2.5) [LSB]	__sigsetjmp(GLIBC_2.2.5) [LSB]	__sysv_signal(GLIBC_2.2.5) [LSB]
bsd_signal(GLIBC_2.2.5) [SUSv3]	psignal(GLIBC_2.2.5) [LSB]	raise(GLIBC_2.2.5) [SUSv3]	sigaction(GLIBC_2.2.5) [SUSv3]
sigaddset(GLIBC_2.2.5) [SUSv3]	sigaltstack(GLIBC_2.2.5) [SUSv3]	sigandset(GLIBC_2.2.5) [LSB]	sigdelset(GLIBC_2.2.5) [SUSv3]
sigemptyset(GLIBC_2.2.5) [SUSv3]	sigfillset(GLIBC_2.2.5) [SUSv3]	sighold(GLIBC_2.2.5) [SUSv3]	sigignore(GLIBC_2.2.5) [SUSv3]
siginterrupt(GLIBC_2.2.5) [SUSv3]	sigisemptyset(GLIBC_2.2.5) [LSB]	sigismember(GLIBC_2.2.5) [SUSv3]	siglongjmp(GLIBC_2.2.5) [SUSv3]
signal(GLIBC_2.2.5) [SUSv3]	sigorset(GLIBC_2.2.5) [LSB]	sigpause(GLIBC_2.2.5) [SUSv3]	sigpending(GLIBC_2.2.5) [SUSv3]
sigprocmask(GLIBC_2.2.5) [SUSv3]	sigqueue(GLIBC_2.2.5) [SUSv3]	sigrelse(GLIBC_2.2.5) [SUSv3]	sigreturn(GLIBC_2.2.5) [LSB]
sigset(GLIBC_2.2.5) [SUSv3]	sigsuspend(GLIBC_2.2.5) [SUSv3]	sigtimedwait(GLIBC_2.2.5) [SUSv3]	sigwait(GLIBC_2.2.5) [SUSv3]
sigwaitinfo(GLIBC_2.2.5) [SUSv3]			

42

43

44

45

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

46

Table 11-7 libc - Signal Handling Data Interfaces

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

47

11.2.5 Localization Functions

48

11.2.5.1 Interfaces for Localization Functions

49

50

51

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

52

Table 11-8 libc - Localization Functions Function Interfaces

bind_textdomain_codeset(GLIBC_2.2.5) [LSB]	bindtextdomain(GLIBC_2.2.5) [LSB]	catclose(GLIBC_2.2.5) [SUSv3]	catgets(GLIBC_2.2.5) [SUSv3]
catopen(GLIBC_2.2.5) [SUSv3]	dcgettext(GLIBC_2.2.5) [LSB]	dcngettext(GLIBC_2.2.5) [LSB]	dgettext(GLIBC_2.2.5) [LSB]
dcngettext(GLIBC_2.2.5) [LSB]	gettext(GLIBC_2.2.5) [LSB]	iconv(GLIBC_2.2.5) [SUSv3]	iconv_close(GLIBC_2.2.5) [SUSv3]

iconv_open(GLIBC_2.2.5) [SUSv3]	localeconv(GLIBC_2.2.5) [SUSv3]	ngettext(GLIBC_2.2.5) [LSB]	nl_langinfo(GLIBC_2.2.5) [SUSv3]
setlocale(GLIBC_2.2.5) [SUSv3]	textdomain(GLIBC_2.2.5) [LSB]		

53

54

55

56

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

57

Table 11-9 libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr(GLIBC_2.2.5) [LSB]			
-------------------------------------	--	--	--

58

11.2.6 Socket Interface

59

11.2.6.1 Interfaces for Socket Interface

60

61

62

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

63

Table 11-10 libc - Socket Interface Function Interfaces

__h_errno_location(GLIBC_2.2.5) [LSB]	accept(GLIBC_2.2.5) [SUSv3]	bind(GLIBC_2.2.5) [SUSv3]	bindresvport(GLIBC_2.2.5) [LSB]
connect(GLIBC_2.2.5) [SUSv3]	gethostid(GLIBC_2.2.5) [SUSv3]	gethostname(GLIBC_2.2.5) [SUSv3]	getpeername(GLIBC_2.2.5) [SUSv3]
getsockname(GLIBC_2.2.5) [SUSv3]	getsockopt(GLIBC_2.2.5) [LSB]	if_freenameindex(GLIBC_2.2.5) [SUSv3]	if_indextoname(GLIBC_2.2.5) [SUSv3]
if_nameindex(GLIBC_2.2.5) [SUSv3]	if_nametoindex(GLIBC_2.2.5) [SUSv3]	listen(GLIBC_2.2.5) [SUSv3]	recv(GLIBC_2.2.5) [SUSv3]
recvfrom(GLIBC_2.2.5) [SUSv3]	recvmsg(GLIBC_2.2.5) [SUSv3]	send(GLIBC_2.2.5) [SUSv3]	sendmsg(GLIBC_2.2.5) [SUSv3]
sendto(GLIBC_2.2.5) [SUSv3]	setsockopt(GLIBC_2.2.5) [LSB]	shutdown(GLIBC_2.2.5) [SUSv3]	socketatmark(GLIBC_2.2.5) [SUSv3]
socket(GLIBC_2.2.5) [SUSv3]	socketpair(GLIBC_2.2.5) [SUSv3]		

64

11.2.7 Wide Characters

65

11.2.7.1 Interfaces for Wide Characters

66

67

68

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

Table 11-11 libc - Wide Characters Function Interfaces

<code>__wcstod_internal</code> (GLIBC_2.2.5) [LSB]	<code>__wcstof_internal</code> (GLIBC_2.2.5) [LSB]	<code>__wcstol_internal</code> (GLIBC_2.2.5) [LSB]	<code>__wcstold_interna</code> <code>l</code> (GLIBC_2.2.5) [LSB]
<code>__wcstoul_interna</code> <code>l</code> (GLIBC_2.2.5) [LSB]	<code>btowc</code> (GLIBC_2.2.5) [SUSv3]	<code>fgetwc</code> (GLIBC_2.2.5) [SUSv3]	<code>fgetws</code> (GLIBC_2.2.5) [SUSv3]
<code>fputwc</code> (GLIBC_2.2.5) [SUSv3]	<code>fputws</code> (GLIBC_2.2.5) [SUSv3]	<code>fwide</code> (GLIBC_2.2.5) [SUSv3]	<code>fwprintf</code> (GLIBC_2.2.5) [SUSv3]
<code>fwscanf</code> (GLIBC_2.2.5) [LSB]	<code>getwc</code> (GLIBC_2.2.5) [SUSv3]	<code>getwchar</code> (GLIBC_2.2.5) [SUSv3]	<code>mblen</code> (GLIBC_2.2.5) [SUSv3]
<code>mbrlen</code> (GLIBC_2.2.5) [SUSv3]	<code>mbrtowc</code> (GLIBC_2.2.5) [SUSv3]	<code>mbsinit</code> (GLIBC_2.2.5) [SUSv3]	<code>mbsnrtowcs</code> (GLIBC_2.2.5) [LSB]
<code>mbsrtowcs</code> (GLIBC_2.2.5) [SUSv3]	<code>mbstowcs</code> (GLIBC_2.2.5) [SUSv3]	<code>mbtowc</code> (GLIBC_2.2.5) [SUSv3]	<code>putwc</code> (GLIBC_2.2.5) [SUSv3]
<code>putwchar</code> (GLIBC_2.2.5) [SUSv3]	<code>swprintf</code> (GLIBC_2.2.5) [SUSv3]	<code>swscanf</code> (GLIBC_2.2.5) [LSB]	<code>towctrans</code> (GLIBC_2.2.5) [SUSv3]
<code>towlower</code> (GLIBC_2.2.5) [SUSv3]	<code>towupper</code> (GLIBC_2.2.5) [SUSv3]	<code>ungetwc</code> (GLIBC_2.2.5) [SUSv3]	<code>vfwprintf</code> (GLIBC_2.2.5) [SUSv3]
<code>vfwscanf</code> (GLIBC_2.2.5) [LSB]	<code>vswprintf</code> (GLIBC_2.2.5) [SUSv3]	<code>vswscanf</code> (GLIBC_2.2.5) [LSB]	<code>vwprintf</code> (GLIBC_2.2.5) [SUSv3]
<code>vwscanf</code> (GLIBC_2.2.5) [LSB]	<code>wcpcpy</code> (GLIBC_2.2.5) [LSB]	<code>wcpncpy</code> (GLIBC_2.2.5) [LSB]	<code>wcrtomb</code> (GLIBC_2.2.5) [SUSv3]
<code>wcscasecmp</code> (GLIBC_2.2.5) [LSB]	<code>wcscat</code> (GLIBC_2.2.5) [SUSv3]	<code>wcschr</code> (GLIBC_2.2.5) [SUSv3]	<code>wcscmp</code> (GLIBC_2.2.5) [SUSv3]
<code>wscoll</code> (GLIBC_2.2.5) [SUSv3]	<code>wscpy</code> (GLIBC_2.2.5) [SUSv3]	<code>wcscspn</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsdup</code> (GLIBC_2.2.5) [LSB]
<code>wcsftime</code> (GLIBC_2.2.5) [SUSv3]	<code>wcslen</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsncasecmp</code> (GLIBC_2.2.5) [LSB]	<code>wcsncat</code> (GLIBC_2.2.5) [SUSv3]
<code>wcsncmp</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsncpy</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsnlen</code> (GLIBC_2.2.5) [LSB]	<code>wcsnrtombs</code> (GLIBC_2.2.5) [LSB]
<code>wcspbrk</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsrchr</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsrtombs</code> (GLIBC_2.2.5) [SUSv3]	<code>wcsspn</code> (GLIBC_2.2.5) [SUSv3]
<code>wcsstr</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstod</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstof</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstoimax</code> (GLIBC_2.2.5) [SUSv3]
<code>wcstok</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstol</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstold</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstoll</code> (GLIBC_2.2.5) [SUSv3]
<code>wcstombs</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstoq</code> (GLIBC_2.2.5) [LSB]	<code>wcstoul</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstoull</code> (GLIBC_2.2.5) [SUSv3]
<code>wcstoumax</code> (GLIBC_2.2.5) [SUSv3]	<code>wcstouq</code> (GLIBC_2.2.5) [LSB]	<code>wcswcs</code> (GLIBC_2.2.5) [SUSv3]	<code>wcswidth</code> (GLIBC_2.2.5) [SUSv3]

wcsxfrm(GLIBC_2.2.5) [SUSv3]	wctob(GLIBC_2.2.5) [SUSv3]	wctomb(GLIBC_2.2.5) [SUSv3]	wctrans(GLIBC_2.2.5) [SUSv3]
wctype(GLIBC_2.2.5) [SUSv3]	wcwidth(GLIBC_2.2.5) [SUSv3]	wmemchr(GLIBC_2.2.5) [SUSv3]	wmemcmp(GLIBC_2.2.5) [SUSv3]
wmemcpy(GLIBC_2.2.5) [SUSv3]	wmemmove(GLIBC_2.2.5) [SUSv3]	wmemset(GLIBC_2.2.5) [SUSv3]	wprintf(GLIBC_2.2.5) [SUSv3]
wscanf(GLIBC_2.2.5) [LSB]			

70

11.2.8 String Functions

71

11.2.8.1 Interfaces for String Functions

72

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

73

74

75

Table 11-12 libc - String Functions Function Interfaces

__memcpy(GLIBC_2.2.5) [LSB]	__rawmemchr(GLIBC_2.2.5) [LSB]	__stpcpy(GLIBC_2.2.5) [LSB]	__strdup(GLIBC_2.2.5) [LSB]
__strtod_internal(GLIBC_2.2.5) [LSB]	__strtof_internal(GLIBC_2.2.5) [LSB]	__strtok_r(GLIBC_2.2.5) [LSB]	__strtol_internal(GLIBC_2.2.5) [LSB]
__strtol_internal(GLIBC_2.2.5) [LSB]	__strtoll_internal(GLIBC_2.2.5) [LSB]	__strtoul_internal(GLIBC_2.2.5) [LSB]	__strtoull_internal(GLIBC_2.2.5) [LSB]
bcmp(GLIBC_2.2.5) [SUSv3]	bcopy(GLIBC_2.2.5) [SUSv3]	bzero(GLIBC_2.2.5) [SUSv3]	ffs(GLIBC_2.2.5) [SUSv3]
index(GLIBC_2.2.5) [SUSv3]	memccpy(GLIBC_2.2.5) [SUSv3]	memchr(GLIBC_2.2.5) [SUSv3]	memcmp(GLIBC_2.2.5) [SUSv3]
memcpy(GLIBC_2.2.5) [SUSv3]	memmove(GLIBC_2.2.5) [SUSv3]	memrchr(GLIBC_2.2.5) [LSB]	memset(GLIBC_2.2.5) [SUSv3]
rindex(GLIBC_2.2.5) [SUSv3]	stpcpy(GLIBC_2.2.5) [LSB]	stpncpy(GLIBC_2.2.5) [LSB]	strcasemp(GLIBC_2.2.5) [SUSv3]
strcasestr(GLIBC_2.2.5) [LSB]	strcat(GLIBC_2.2.5) [SUSv3]	strchr(GLIBC_2.2.5) [SUSv3]	strcmp(GLIBC_2.2.5) [SUSv3]
strcoll(GLIBC_2.2.5) [SUSv3]	strcpy(GLIBC_2.2.5) [SUSv3]	strcspn(GLIBC_2.2.5) [SUSv3]	strdup(GLIBC_2.2.5) [SUSv3]
strerror(GLIBC_2.2.5) [SUSv3]	strerror_r(GLIBC_2.2.5) [LSB]	strfmon(GLIBC_2.2.5) [SUSv3]	strftime(GLIBC_2.2.5) [SUSv3]
strlen(GLIBC_2.2.5) [SUSv3]	strncasemp(GLIBC_2.2.5) [SUSv3]	strncat(GLIBC_2.2.5) [SUSv3]	strncmp(GLIBC_2.2.5) [SUSv3]
strncpy(GLIBC_2.2.5) [SUSv3]	strndup(GLIBC_2.2.5) [LSB]	strnlen(GLIBC_2.2.5) [LSB]	strpbrk(GLIBC_2.2.5) [SUSv3]

strptime(GLIBC_2.2.5) [LSB]	strchr(GLIBC_2.2.5) [SUSv3]	strsep(GLIBC_2.2.5) [LSB]	strsignal(GLIBC_2.2.5) [LSB]
strspn(GLIBC_2.2.5) [SUSv3]	strstr(GLIBC_2.2.5) [SUSv3]	strtof(GLIBC_2.2.5) [SUSv3]	strtoimax(GLIBC_2.2.5) [SUSv3]
strtok(GLIBC_2.2.5) [SUSv3]	strtok_r(GLIBC_2.2.5) [SUSv3]	strtold(GLIBC_2.2.5) [SUSv3]	strtoll(GLIBC_2.2.5) [SUSv3]
strtoq(GLIBC_2.2.5) [LSB]	strtoull(GLIBC_2.2.5) [SUSv3]	strtoumax(GLIBC_2.2.5) [SUSv3]	strtouq(GLIBC_2.2.5) [LSB]
strxfrm(GLIBC_2.2.5) [SUSv3]	swab(GLIBC_2.2.5) [SUSv3]		

76

11.2.9 IPC Functions

77

11.2.9.1 Interfaces for IPC Functions

78

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

79

80

81

Table 11-13 libc - IPC Functions Function Interfaces

ftok(GLIBC_2.2.5) [SUSv3]	msgctl(GLIBC_2.2.5) [SUSv3]	msgget(GLIBC_2.2.5) [SUSv3]	msgrcv(GLIBC_2.2.5) [SUSv3]
msgsnd(GLIBC_2.2.5) [SUSv3]	semctl(GLIBC_2.2.5) [SUSv3]	semget(GLIBC_2.2.5) [SUSv3]	semop(GLIBC_2.2.5) [SUSv3]
shmat(GLIBC_2.2.5) [SUSv3]	shmctl(GLIBC_2.2.5) [SUSv3]	shmdt(GLIBC_2.2.5) [SUSv3]	shmget(GLIBC_2.2.5) [SUSv3]

82

11.2.10 Regular Expressions

83

11.2.10.1 Interfaces for Regular Expressions

84

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

85

86

87

Table 11-14 libc - Regular Expressions Function Interfaces

regcomp(GLIBC_2.2.5) [SUSv3]	regerror(GLIBC_2.2.5) [SUSv3]	regexexec(GLIBC_2.3.4) [LSB]	regfree(GLIBC_2.2.5) [SUSv3]
------------------------------	-------------------------------	------------------------------	------------------------------

88

11.2.11 Character Type Functions

89

11.2.11.1 Interfaces for Character Type Functions

90

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

91

92

93

Table 11-15 libc - Character Type Functions Function Interfaces

__ctype_get_mb_c	_tolower(GLIBC_2.2.5) [SUSv3]	_toupper(GLIBC_2.2.5) [SUSv3]	isalnum(GLIBC_2.2.5) [SUSv3]
------------------	-------------------------------	-------------------------------	------------------------------

ur_max(GLIBC_2.2.5) [LSB]	2.2.5) [SUSv3]	2.2.5) [SUSv3]	2.5) [SUSv3]
isalpha(GLIBC_2.2.5) [SUSv3]	isascii(GLIBC_2.2.5) [SUSv3]	iscntrl(GLIBC_2.2.5) [SUSv3]	isdigit(GLIBC_2.2.5) [SUSv3]
isgraph(GLIBC_2.2.5) [SUSv3]	islower(GLIBC_2.2.5) [SUSv3]	isprint(GLIBC_2.2.5) [SUSv3]	ispunct(GLIBC_2.2.5) [SUSv3]
isspace(GLIBC_2.2.5) [SUSv3]	isupper(GLIBC_2.2.5) [SUSv3]	iswalnum(GLIBC_2.2.5) [SUSv3]	iswalpha(GLIBC_2.2.5) [SUSv3]
iswblank(GLIBC_2.2.5) [SUSv3]	iswcntrl(GLIBC_2.2.5) [SUSv3]	iswctype(GLIBC_2.2.5) [SUSv3]	iswdigit(GLIBC_2.2.5) [SUSv3]
iswgraph(GLIBC_2.2.5) [SUSv3]	iswlower(GLIBC_2.2.5) [SUSv3]	iswprint(GLIBC_2.2.5) [SUSv3]	iswpunct(GLIBC_2.2.5) [SUSv3]
iswspace(GLIBC_2.2.5) [SUSv3]	iswupper(GLIBC_2.2.5) [SUSv3]	iswxdigit(GLIBC_2.2.5) [SUSv3]	isxdigit(GLIBC_2.2.5) [SUSv3]
toascii(GLIBC_2.2.5) [SUSv3]	tolower(GLIBC_2.2.5) [SUSv3]	toupper(GLIBC_2.2.5) [SUSv3]	

94

11.2.12 Time Manipulation

95

11.2.12.1 Interfaces for Time Manipulation

96

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

97

98

Table 11-16 libc - Time Manipulation Function Interfaces

adjtime(GLIBC_2.2.5) [LSB]	asctime(GLIBC_2.2.5) [SUSv3]	asctime_r(GLIBC_2.2.5) [SUSv3]	ctime(GLIBC_2.2.5) [SUSv3]
ctime_r(GLIBC_2.2.5) [SUSv3]	difftime(GLIBC_2.2.5) [SUSv3]	gmtime(GLIBC_2.2.5) [SUSv3]	gmtime_r(GLIBC_2.2.5) [SUSv3]
localtime(GLIBC_2.2.5) [SUSv3]	localtime_r(GLIBC_2.2.5) [SUSv3]	mktime(GLIBC_2.2.5) [SUSv3]	tzset(GLIBC_2.2.5) [SUSv3]
ualarm(GLIBC_2.2.5) [SUSv3]			

100

101

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

102

103

Table 11-17 libc - Time Manipulation Data Interfaces

__daylight(GLIBC_2.2.5) [LSB]	__timezone(GLIBC_2.2.5) [LSB]	__tzname(GLIBC_2.2.5) [LSB]	daylight(GLIBC_2.2.5) [SUSv3]
timezone(GLIBC_2.2.5) [SUSv3]	tzname(GLIBC_2.2.5) [SUSv3]		

105

11.2.13 Terminal Interface Functions

106

11.2.13.1 Interfaces for Terminal Interface Functions

107

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

108

109

110

Table 11-18 libc - Terminal Interface Functions Function Interfaces

cfgetispeed(GLIBC_2.2.5) [SUSv3]	cfgetospeed(GLIBC_2.2.5) [SUSv3]	cfmakeraw(GLIBC_2.2.5) [LSB]	cfsetispeed(GLIBC_2.2.5) [SUSv3]
cfsetospeed(GLIBC_2.2.5) [SUSv3]	cfsetspeed(GLIBC_2.2.5) [LSB]	tcdrain(GLIBC_2.2.5) [SUSv3]	tcflow(GLIBC_2.2.5) [SUSv3]
tcflush(GLIBC_2.2.5) [SUSv3]	tcgetattr(GLIBC_2.2.5) [SUSv3]	tcgetpgrp(GLIBC_2.2.5) [SUSv3]	tcgetsid(GLIBC_2.2.5) [SUSv3]
tcsendbreak(GLIBC_2.2.5) [SUSv3]	tcsetattr(GLIBC_2.2.5) [SUSv3]	tcsetpgrp(GLIBC_2.2.5) [SUSv3]	

111

11.2.14 System Database Interface

112

11.2.14.1 Interfaces for System Database Interface

113

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

114

115

116

Table 11-19 libc - System Database Interface Function Interfaces

endgrent(GLIBC_2.2.5) [SUSv3]	endprotoent(GLIBC_2.2.5) [SUSv3]	endpwent(GLIBC_2.2.5) [SUSv3]	endservent(GLIBC_2.2.5) [SUSv3]
endutent(GLIBC_2.2.5) [SUSv2]	endutxent(GLIBC_2.2.5) [SUSv3]	getgrent(GLIBC_2.2.5) [SUSv3]	getgrgid(GLIBC_2.2.5) [SUSv3]
getgrgid_r(GLIBC_2.2.5) [SUSv3]	getgrnam(GLIBC_2.2.5) [SUSv3]	getgrnam_r(GLIBC_2.2.5) [SUSv3]	getgrouplist(GLIBC_2.2.5) [LSB]
gethostbyaddr(GLIBC_2.2.5) [SUSv3]	gethostbyname(GLIBC_2.2.5) [SUSv3]	getprotobyname(GLIBC_2.2.5) [SUSv3]	getprotobynumber(GLIBC_2.2.5) [SUSv3]
getprotoent(GLIBC_2.2.5) [SUSv3]	getpwent(GLIBC_2.2.5) [SUSv3]	getpwnam(GLIBC_2.2.5) [SUSv3]	getpwnam_r(GLIBC_2.2.5) [SUSv3]
getpwuid(GLIBC_2.2.5) [SUSv3]	getpwuid_r(GLIBC_2.2.5) [SUSv3]	getservbyname(GLIBC_2.2.5) [SUSv3]	getservbyport(GLIBC_2.2.5) [SUSv3]
getservent(GLIBC_2.2.5) [SUSv3]	getutent(GLIBC_2.2.5) [LSB]	getutent_r(GLIBC_2.2.5) [LSB]	getutxent(GLIBC_2.2.5) [SUSv3]
getutxid(GLIBC_2.2.5) [SUSv3]	getutxline(GLIBC_2.2.5) [SUSv3]	pututxline(GLIBC_2.2.5) [SUSv3]	setgrent(GLIBC_2.2.5) [SUSv3]
setgroups(GLIBC_2.2.5) [SUSv3]	setprotoent(GLIBC_2.2.5) [SUSv3]	setpwent(GLIBC_2.2.5) [SUSv3]	setservent(GLIBC_2.2.5) [SUSv3]

__2.2.5) [LSB]	C_2.2.5) [SUSv3]	2.2.5) [SUSv3]	__2.2.5) [SUSv3]
setutent(GLIBC_2.2.5) [LSB]	setutxent(GLIBC_2.2.5) [SUSv3]	utmpname(GLIBC_2.2.5) [LSB]	

117

11.2.15 Language Support

118

11.2.15.1 Interfaces for Language Support

119

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

120

121

122

Table 11-20 libc - Language Support Function Interfaces

__libc_start_main(GLIBC_2.2.5) [LSB]			
--------------------------------------	--	--	--

123

11.2.16 Large File Support

124

11.2.16.1 Interfaces for Large File Support

125

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

126

127

128

Table 11-21 libc - Large File Support Function Interfaces

__fxstat64(GLIBC_2.2.5) [LSB]	__lxstat64(GLIBC_2.2.5) [LSB]	__xstat64(GLIBC_2.2.5) [LSB]	creat64(GLIBC_2.2.5) [LFS]
fgetpos64(GLIBC_2.2.5) [LFS]	fopen64(GLIBC_2.2.5) [LFS]	freopen64(GLIBC_2.2.5) [LFS]	fseeko64(GLIBC_2.2.5) [LFS]
fsetpos64(GLIBC_2.2.5) [LFS]	fstatvfs64(GLIBC_2.2.5) [LFS]	ftello64(GLIBC_2.2.5) [LFS]	ftruncate64(GLIBC_2.2.5) [LFS]
ftw64(GLIBC_2.2.5) [LFS]	getrlimit64(GLIBC_2.2.5) [LFS]	lockf64(GLIBC_2.2.5) [LFS]	mkstemp64(GLIBC_2.2.5) [LFS]
mmap64(GLIBC_2.2.5) [LFS]	nftw64(GLIBC_2.3) [LFS]	readdir64(GLIBC_2.2.5) [LFS]	statvfs64(GLIBC_2.2.5) [LFS]
tmpfile64(GLIBC_2.2.5) [LFS]	truncate64(GLIBC_2.2.5) [LFS]		

129

11.2.17 Standard Library

130

11.2.17.1 Interfaces for Standard Library

131

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

132

133

134

Table 11-22 libc - Standard Library Function Interfaces

__Exit(GLIBC_2.2.5)	__assert_fail(GLIBC_2.2.5)	__cxa_atexit(GLIBC_2.2.5)	__errno_location(GLIBC_2.2.5)
---------------------	----------------------------	---------------------------	-------------------------------

) [SUSv3]	C_2.2.5) [LSB]	C_2.2.5) [LSB]	GLIBC_2.2.5) [LSB]
__fpending(GLIBC_2.2.5) [LSB]	__getpagesize(GLIBC_2.2.5) [LSB]	__isinf(GLIBC_2.2.5) [LSB]	__isinf(GLIBC_2.2.5) [LSB]
__isinfl(GLIBC_2.2.5) [LSB]	__isnan(GLIBC_2.2.5) [LSB]	__isnanf(GLIBC_2.2.5) [LSB]	__isnanl(GLIBC_2.2.5) [LSB]
__sysconf(GLIBC_2.2.5) [LSB]	_exit(GLIBC_2.2.5) [SUSv3]	_longjmp(GLIBC_2.2.5) [SUSv3]	_setjmp(GLIBC_2.2.5) [SUSv3]
a64l(GLIBC_2.2.5) [SUSv3]	abort(GLIBC_2.2.5) [SUSv3]	abs(GLIBC_2.2.5) [SUSv3]	atof(GLIBC_2.2.5) [SUSv3]
atoi(GLIBC_2.2.5) [SUSv3]	atol(GLIBC_2.2.5) [SUSv3]	atoll(GLIBC_2.2.5) [SUSv3]	basename(GLIBC_2.2.5) [SUSv3]
bsearch(GLIBC_2.2.5) [SUSv3]	calloc(GLIBC_2.2.5) [SUSv3]	closelog(GLIBC_2.2.5) [SUSv3]	confstr(GLIBC_2.2.5) [SUSv3]
cuserid(GLIBC_2.2.5) [SUSv2]	daemon(GLIBC_2.2.5) [LSB]	dirname(GLIBC_2.2.5) [SUSv3]	div(GLIBC_2.2.5) [SUSv3]
drand48(GLIBC_2.2.5) [SUSv3]	ecvt(GLIBC_2.2.5) [SUSv3]	erand48(GLIBC_2.2.5) [SUSv3]	err(GLIBC_2.2.5) [LSB]
error(GLIBC_2.2.5) [LSB]	errx(GLIBC_2.2.5) [LSB]	fcvt(GLIBC_2.2.5) [SUSv3]	fmtmsg(GLIBC_2.2.5) [SUSv3]
fnmatch(GLIBC_2.2.5) [SUSv3]	fpathconf(GLIBC_2.2.5) [SUSv3]	free(GLIBC_2.2.5) [SUSv3]	freaddrinfo(GLIBC_2.2.5) [SUSv3]
ftrylockfile(GLIBC_2.2.5) [SUSv3]	ftw(GLIBC_2.2.5) [SUSv3]	funlockfile(GLIBC_2.2.5) [SUSv3]	gai_strerror(GLIBC_2.2.5) [SUSv3]
gcvt(GLIBC_2.2.5) [SUSv3]	getaddrinfo(GLIBC_2.2.5) [SUSv3]	getcwd(GLIBC_2.2.5) [SUSv3]	getdate(GLIBC_2.2.5) [SUSv3]
getenv(GLIBC_2.2.5) [SUSv3]	getlogin(GLIBC_2.2.5) [SUSv3]	getlogin_r(GLIBC_2.2.5) [SUSv3]	getnameinfo(GLIBC_2.2.5) [SUSv3]
getopt(GLIBC_2.2.5) [LSB]	getopt_long(GLIBC_2.2.5) [LSB]	getopt_long_only(GLIBC_2.2.5) [LSB]	getsubopt(GLIBC_2.2.5) [SUSv3]
gettimeofday(GLIBC_2.2.5) [SUSv3]	glob(GLIBC_2.2.5) [SUSv3]	glob64(GLIBC_2.2.5) [LSB]	globfree(GLIBC_2.2.5) [SUSv3]
globfree64(GLIBC_2.2.5) [LSB]	grantpt(GLIBC_2.2.5) [SUSv3]	hcreate(GLIBC_2.2.5) [SUSv3]	hdestroy(GLIBC_2.2.5) [SUSv3]
hsearch(GLIBC_2.2.5) [SUSv3]	htonl(GLIBC_2.2.5) [SUSv3]	htons(GLIBC_2.2.5) [SUSv3]	imaxabs(GLIBC_2.2.5) [SUSv3]
imaxdiv(GLIBC_2.2.5) [SUSv3]	inet_addr(GLIBC_2.2.5) [SUSv3]	inet_ntoa(GLIBC_2.2.5) [SUSv3]	inet_ntop(GLIBC_2.2.5) [SUSv3]
inet_pton(GLIBC_2.2.5) [SUSv3]	initstate(GLIBC_2.2.5) [SUSv3]	insque(GLIBC_2.2.5) [SUSv3]	isatty(GLIBC_2.2.5) [SUSv3]

isblank(GLIBC_2.2.5) [SUSv3]	jrand48(GLIBC_2.2.5) [SUSv3]	l64a(GLIBC_2.2.5) [SUSv3]	labs(GLIBC_2.2.5) [SUSv3]
lcong48(GLIBC_2.2.5) [SUSv3]	ldiv(GLIBC_2.2.5) [SUSv3]	lfind(GLIBC_2.2.5) [SUSv3]	llabs(GLIBC_2.2.5) [SUSv3]
lldiv(GLIBC_2.2.5) [SUSv3]	longjmp(GLIBC_2.2.5) [SUSv3]	lrnd48(GLIBC_2.2.5) [SUSv3]	lsearch(GLIBC_2.2.5) [SUSv3]
makecontext(GLIBC_2.2.5) [SUSv3]	malloc(GLIBC_2.2.5) [SUSv3]	memmem(GLIBC_2.2.5) [LSB]	mkstemp(GLIBC_2.2.5) [SUSv3]
mktemp(GLIBC_2.2.5) [SUSv3]	mrnd48(GLIBC_2.2.5) [SUSv3]	nftw(GLIBC_2.3.3) [SUSv3]	nrnd48(GLIBC_2.2.5) [SUSv3]
ntohl(GLIBC_2.2.5) [SUSv3]	ntohs(GLIBC_2.2.5) [SUSv3]	openlog(GLIBC_2.2.5) [SUSv3]	perror(GLIBC_2.2.5) [SUSv3]
posix_memalign(GLIBC_2.2.5) [SUSv3]	posix_openpt(GLIBC_2.2.5) [SUSv3]	ptsname(GLIBC_2.2.5) [SUSv3]	putenv(GLIBC_2.2.5) [SUSv3]
qsort(GLIBC_2.2.5) [SUSv3]	rand(GLIBC_2.2.5) [SUSv3]	rand_r(GLIBC_2.2.5) [SUSv3]	random(GLIBC_2.2.5) [SUSv3]
realloc(GLIBC_2.2.5) [SUSv3]	realpath(GLIBC_2.2.5) [SUSv3]	remque(GLIBC_2.2.5) [SUSv3]	seed48(GLIBC_2.2.5) [SUSv3]
setenv(GLIBC_2.2.5) [SUSv3]	sethostname(GLIBC_2.2.5) [LSB]	setlogmask(GLIBC_2.2.5) [SUSv3]	setstate(GLIBC_2.2.5) [SUSv3]
srand(GLIBC_2.2.5) [SUSv3]	srand48(GLIBC_2.2.5) [SUSv3]	srandom(GLIBC_2.2.5) [SUSv3]	strtod(GLIBC_2.2.5) [SUSv3]
strtol(GLIBC_2.2.5) [SUSv3]	strtoul(GLIBC_2.2.5) [SUSv3]	swapcontext(GLIBC_2.2.5) [SUSv3]	syslog(GLIBC_2.2.5) [SUSv3]
system(GLIBC_2.2.5) [LSB]	tdelete(GLIBC_2.2.5) [SUSv3]	tfind(GLIBC_2.2.5) [SUSv3]	tmpfile(GLIBC_2.2.5) [SUSv3]
tmpnam(GLIBC_2.2.5) [SUSv3]	tsearch(GLIBC_2.2.5) [SUSv3]	ttyname(GLIBC_2.2.5) [SUSv3]	ttyname_r(GLIBC_2.2.5) [SUSv3]
twalk(GLIBC_2.2.5) [SUSv3]	unlockpt(GLIBC_2.2.5) [SUSv3]	unsetenv(GLIBC_2.2.5) [SUSv3]	usleep(GLIBC_2.2.5) [SUSv3]
verrx(GLIBC_2.2.5) [LSB]	vfscanf(GLIBC_2.2.5) [LSB]	vscanf(GLIBC_2.2.5) [LSB]	vsscanf(GLIBC_2.2.5) [LSB]
vsyslog(GLIBC_2.2.5) [LSB]	warn(GLIBC_2.2.5) [LSB]	warnx(GLIBC_2.2.5) [LSB]	wordexp(GLIBC_2.2.5) [SUSv3]
wordfree(GLIBC_2.2.5) [SUSv3]			

135

136

137

138

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

139

Table 11-23 libc - Standard Library Data Interfaces

<code>__environ</code> (GLIBC_2.2.5) [LSB]	<code>_environ</code> (GLIBC_2.2.5) [LSB]	<code>_sys_errlist</code> (GLIBC_2.3) [LSB]	<code>environ</code> (GLIBC_2.2.5) [SUSv3]
<code>getdate_err</code> (GLIBC_2.2.5) [SUSv3]	<code>optarg</code> (GLIBC_2.2.5) [SUSv3]	<code>opterr</code> (GLIBC_2.2.5) [SUSv3]	<code>optind</code> (GLIBC_2.2.5) [SUSv3]
<code>optopt</code> (GLIBC_2.2.5) [SUSv3]			

140

11.3 Data Definitions for libc

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

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

151 This specification uses the ISO C (1999) C Language as the reference programming
 152 language, and data definitions are specified in ISO C format. The C language is used
 153 here as a convenient notation. Using a C language description of these data objects
 154 does not preclude their use by other programming languages.

11.3.1 arpa/inet.h

```
155
156 extern uint32_t htonl(uint32_t);
157 extern uint16_t htons(uint16_t);
158 extern in_addr_t inet_addr(const char *);
159 extern char *inet_ntoa(struct in_addr);
160 extern const char *inet_ntop(int, const void *, char *, socklen_t);
161 extern int inet_pton(int, const char *, void *);
162 extern uint32_t ntohl(uint32_t);
163 extern uint16_t ntohs(uint16_t);
```

11.3.2 assert.h

```
164
165 extern void __assert_fail(const char *, const char *, unsigned int,
166                          const char *);
```

11.3.3 ctype.h

```
167
168 extern int _tolower(int);
169 extern int _toupper(int);
170 extern int isalnum(int);
171 extern int isalpha(int);
172 extern int isascii(int);
173 extern int iscntrl(int);
174 extern int isdigit(int);
175 extern int isgraph(int);
176 extern int islower(int);
```



```

177     extern int isprint(int);
178     extern int ispunct(int);
179     extern int isspace(int);
180     extern int isupper(int);
181     extern int isxdigit(int);
182     extern int toascii(int);
183     extern int tolower(int);
184     extern int toupper(int);
185     extern int isblank(int);
186     extern const unsigned short **__ctype_b_loc(void);
187     extern const int32_t **__ctype_toupper_loc(void);
188     extern const int32_t **__ctype_tolower_loc(void);

```

11.3.4 dirent.h

```

189
190     extern void rewinddir(DIR *);
191     extern void seekdir(DIR *, long int);
192     extern long int telldir(DIR *);
193     extern int closedir(DIR *);
194     extern DIR *opendir(const char *);
195     extern struct dirent *readdir(DIR *);
196     extern struct dirent64 *readdir64(DIR *);
197     extern int readdir_r(DIR *, struct dirent *, struct dirent **);

```

11.3.5 err.h

```

198
199     extern void err(int, const char *, ...);
200     extern void errx(int, const char *, ...);
201     extern void warn(const char *, ...);
202     extern void warnx(const char *, ...);
203     extern void error(int, int, const char *, ...);

```

11.3.6 errno.h

```

204
205     #define EDEADLOCK          EDEADLK
206
207     extern int *__errno_location(void);

```

11.3.7 fcntl.h

```

208
209     #define F_GETLK64          5
210     #define F_SETLK64          6
211     #define F_SETLKW64         7
212
213     extern int lockf64(int, int, off64_t);
214     extern int fcntl(int, int, ...);

```

11.3.8 fmtmsg.h

```

215
216     extern int fmtmsg(long int, const char *, int, const char *, const char
217 *,
218                      const char *);

```

11.3.9 fnmatch.h

```

219
220     extern int fnmatch(const char *, const char *, int);

```

11.3.10 ftw.h

```

221
222 extern int ftw(const char *, __ftw_func_t, int);
223 extern int ftw64(const char *, __ftw64_func_t, int);
224 extern int nftw(const char *, __nftw_func_t, int, int);
225 extern int nftw64(const char *, __nftw64_func_t, int, int);

```

11.3.11 getopt.h

```

226
227 extern int getopt_long(int, char *const, const char *,
228                       const struct option *, int *);
229 extern int getopt_long_only(int, char *const, const char *,
230                             const struct option *, int *);

```

11.3.12 glob.h

```

231
232 extern int glob(const char *, int,
233               int (*__errfunc) (const char *p1, int p2)
234               , glob_t *);
235 extern int glob64(const char *, int,
236                 int (*__errfunc) (const char *p1, int p2)
237                 , glob64_t *);
238 extern void globfree(glob_t *);
239 extern void globfree64(glob64_t *);

```

11.3.13 grp.h

```

240
241 extern void endgrent(void);
242 extern struct group *getgrent(void);
243 extern struct group *getgrgid(gid_t);
244 extern struct group *getgrnam(char *);
245 extern int initgroups(const char *, gid_t);
246 extern void setgrent(void);
247 extern int setgroups(size_t, const gid_t *);
248 extern int getgrgid_r(gid_t, struct group *, char *, size_t,
249                     struct group **);
250 extern int getgrnam_r(const char *, struct group *, char *, size_t,
251                      struct group **);
252 extern int getgrouplist(const char *, gid_t, gid_t *, int *);

```

11.3.14 iconv.h

```

253
254 extern size_t iconv(iconv_t, char **, size_t *, char **, size_t *);
255 extern int iconv_close(iconv_t);
256 extern iconv_t iconv_open(char *, char *);

```

11.3.15 inttypes.h

```

257
258 typedef long int intmax_t;
259 typedef unsigned long int uintptr_t;
260 typedef unsigned long int uintmax_t;
261 typedef unsigned long int uint64_t;
262
263 extern intmax_t strtoumax(const char *, char **, int);
264 extern uintmax_t strtoumax(const char *, char **, int);
265 extern intmax_t wcstoumax(const wchar_t *, wchar_t * *, int);

```

```

266     extern uintmax_t wcstoumax(const wchar_t *, wchar_t * *, int);
267     extern intmax_t imaxabs(intmax_t);
268     extern imaxdiv_t imaxdiv(intmax_t, intmax_t);

```

11.3.16 langinfo.h

```

269
270     extern char *nl_langinfo(nl_item);

```

11.3.17 libgen.h

```

271
272     extern char *basename(const char *);
273     extern char *dirname(char *);

```

11.3.18 libintl.h

```

274
275     extern char *bindtextdomain(const char *, const char *);
276     extern char *dcgettext(const char *, const char *, int);
277     extern char *dgettext(const char *, const char *);
278     extern char *gettext(const char *);
279     extern char *textdomain(const char *);
280     extern char *bind_textdomain_codeset(const char *, const char *);
281     extern char *dcngettext(const char *, const char *, const char *,
282                             unsigned long int, int);
283     extern char *dncgettext(const char *, const char *, const char *,
284                             unsigned long int);
285     extern char *ngettext(const char *, const char *, unsigned long int);

```

11.3.19 limits.h

```

286
287     #define LONG_MAX          0x7FFFFFFFFFFFFFFFL
288     #define ULONG_MAX        0xFFFFFFFFFFFFFFFFUL
289
290     #define CHAR_MAX         127
291     #define CHAR_MIN         SCHAR_MIN
292
293     #define PTHREAD_STACK_MIN 16384

```

11.3.20 locale.h

```

294
295     extern struct lconv *localeconv(void);
296     extern char *setlocale(int, const char *);
297     extern locale_t uselocale(locale_t);
298     extern void freelocale(locale_t);
299     extern locale_t duplocale(locale_t);
300     extern locale_t newlocale(int, const char *, locale_t);

```

11.3.21 monetary.h

```

301
302     extern ssize_t strfmon(char *, size_t, const char *, ...);

```

11.3.22 net/if.h

```

303
304     extern void if_freenameindex(struct if_nameindex *);
305     extern char *if_indextoname(unsigned int, char *);
306     extern struct if_nameindex *if_nameindex(void);

```

```
307     extern unsigned int if_nametoindex(const char *);
```

11.3.23 netdb.h

```
308
309     extern void endprotoent(void);
310     extern void endservent(void);
311     extern void freeaddrinfo(struct addrinfo *);
312     extern const char *gai_strerror(int);
313     extern int getaddrinfo(const char *, const char *, const struct addrinfo
314     *,
315     struct addrinfo **);
316     extern struct hostent *gethostbyaddr(const void *, socklen_t, int);
317     extern struct hostent *gethostbyname(const char *);
318     extern struct protoent *getprotobyname(const char *);
319     extern struct protoent *getprotobynumber(int);
320     extern struct protoent *getprotoent(void);
321     extern struct servent *getservbyname(const char *, const char *);
322     extern struct servent *getservbyport(int, const char *);
323     extern struct servent *getservent(void);
324     extern void setprotoent(int);
325     extern void setservent(int);
326     extern int *__h_errno_location(void);
```

11.3.24 netinet/in.h

```
327
328     extern int bindresvport(int, struct sockaddr_in *);
```

11.3.25 netinet/ip.h

```
329
330     /*
331     * This header is architecture neutral
332     * Please refer to the generic specification for details
333     */
```

11.3.26 netinet/tcp.h

```
334
335     /*
336     * This header is architecture neutral
337     * Please refer to the generic specification for details
338     */
```

11.3.27 netinet/udp.h

```
339
340     /*
341     * This header is architecture neutral
342     * Please refer to the generic specification for details
343     */
```

11.3.28 nl_types.h

```
344
345     extern int catclose(nl_catd);
346     extern char *catgets(nl_catd, int, int, const char *);
347     extern nl_catd catopen(const char *, int);
```

11.3.29 poll.h

```

348
349 extern int poll(struct pollfd *, nfds_t, int);

```

11.3.30 pty.h

```

350
351 extern int openpty(int *, int *, char *, struct termios *,
352                  struct winsize *);
353 extern int forkpty(int *, char *, struct termios *, struct winsize *);

```

11.3.31 pwd.h

```

354
355 extern void endpwent(void);
356 extern struct passwd *getpwent(void);
357 extern struct passwd *getpwnam(char *);
358 extern struct passwd *getpwuid(uid_t);
359 extern void setpwent(void);
360 extern int getpwnam_r(char *, struct passwd *, char *, size_t,
361                      struct passwd **);
362 extern int getpwuid_r(uid_t, struct passwd *, char *, size_t,
363                      struct passwd **);

```

11.3.32 regex.h

```

364
365 extern int regcomp(regex_t *, const char *, int);
366 extern size_t regerror(int, const regex_t *, char *, size_t);
367 extern int regexec(const regex_t *, const char *, size_t, regmatch_t,
368                   int);
369 extern void regfree(regex_t *);

```

11.3.33 rpc/auth.h

```

370
371 extern struct AUTH *authnone_create(void);
372 extern int key_decryptsession(char *, union des_block *);
373 extern bool_t xdr_opaque_auth(XDR *, struct opaque_auth *);

```

11.3.34 rpc/clnt.h

```

374
375 extern struct CLIENT *clnt_create(const char *, const u_long, const
376 u_long,
377                                 const char *);
378 extern void clnt_pcreateerror(const char *);
379 extern void clnt_perrno(enum clnt_stat);
380 extern void clnt_perror(struct CLIENT *, const char *);
381 extern char *clnt_spcreateerror(const char *);
382 extern char *clnt_sperrno(enum clnt_stat);
383 extern char *clnt_sperror(struct CLIENT *, const char *);

```

11.3.35 rpc/pmap_clnt.h

```

384
385 extern u_short pmap_getport(struct sockaddr_in *, const u_long,
386                             const u_long, u_int);
387 extern bool_t pmap_set(const u_long, const u_long, int, u_short);
388 extern bool_t pmap_unset(u_long, u_long);

```

11.3.36 rpc/rpc_msg.h

```

389
390 extern bool_t xdr_callhdr(XDR *, struct rpc_msg *);

```

11.3.37 rpc/svc.h

```

391
392 extern void svc_getreqset(fd_set *);
393 extern bool_t svc_register(SVCXPRT *, rpcprog_t, rpcvers_t,
394     __dispatch_fn_t, rpcprot_t);
395 extern void svc_run(void);
396 extern bool_t svc_sendreply(SVCXPRT *, xdrproc_t, caddr_t);
397 extern void svcerr_auth(SVCXPRT *, enum auth_stat);
398 extern void svcerr_decode(SVCXPRT *);
399 extern void svcerr_noproc(SVCXPRT *);
400 extern void svcerr_noprogram(SVCXPRT *);
401 extern void svcerr_progvers(SVCXPRT *, rpcvers_t, rpcvers_t);
402 extern void svcerr_systemerr(SVCXPRT *);
403 extern void svcerr_weakauth(SVCXPRT *);
404 extern SVCXPRT *svctcp_create(int, u_int, u_int);
405 extern SVCXPRT *svccudp_create(int);

```

11.3.38 rpc/types.h

```

406
407 /*
408  * This header is architecture neutral
409  * Please refer to the generic specification for details
410  */

```

11.3.39 rpc/xdr.h

```

411
412 extern bool_t xdr_array(XDR *, caddr_t *, u_int *, u_int, u_int,
413     xdrproc_t);
414 extern bool_t xdr_bool(XDR *, bool_t *);
415 extern bool_t xdr_bytes(XDR *, char **, u_int *, u_int);
416 extern bool_t xdr_char(XDR *, char *);
417 extern bool_t xdr_double(XDR *, double *);
418 extern bool_t xdr_enum(XDR *, enum_t *);
419 extern bool_t xdr_float(XDR *, float *);
420 extern void xdr_free(xdrproc_t, char *);
421 extern bool_t xdr_int(XDR *, int *);
422 extern bool_t xdr_long(XDR *, long int *);
423 extern bool_t xdr_opaque(XDR *, caddr_t, u_int);
424 extern bool_t xdr_pointer(XDR *, char **, u_int, xdrproc_t);
425 extern bool_t xdr_reference(XDR *, caddr_t *, u_int, xdrproc_t);
426 extern bool_t xdr_short(XDR *, short *);
427 extern bool_t xdr_string(XDR *, char **, u_int);
428 extern bool_t xdr_u_char(XDR *, u_char *);
429 extern bool_t xdr_u_int(XDR *, u_int *);
430 extern bool_t xdr_u_long(XDR *, u_long *);
431 extern bool_t xdr_u_short(XDR *, u_short *);
432 extern bool_t xdr_union(XDR *, enum_t *, char *,
433     const struct xdr_discrim *, xdrproc_t);
434 extern bool_t xdr_vector(XDR *, char *, u_int, u_int, xdrproc_t);
435 extern bool_t xdr_void(void);
436 extern bool_t xdr_wrapstring(XDR *, char **);
437 extern void xdrmem_create(XDR *, caddr_t, u_int, enum xdr_op);
438 extern void xdrrec_create(XDR *, u_int, u_int, caddr_t,
439     int (*__readit) (char *p1, char *p2, int p3)

```

```

440         , int (*__writeit) (char *p1, char *p2, int
441         p3)
442         );
443 extern typedef int bool_t xdrrec_eof(XDR *);

```

11.3.40 sched.h

```

444
445 extern int sched_get_priority_max(int);
446 extern int sched_get_priority_min(int);
447 extern int sched_getparam(pid_t, struct sched_param *);
448 extern int sched_getscheduler(pid_t);
449 extern int sched_rr_get_interval(pid_t, struct timespec *);
450 extern int sched_setparam(pid_t, const struct sched_param *);
451 extern int sched_setscheduler(pid_t, int, const struct sched_param *);
452 extern int sched_yield(void);

```

11.3.41 search.h

```

453
454 extern int hcreate(size_t);
455 extern ENTRY *hsearch(ENTRY, ACTION);
456 extern void insque(void *, void *);
457 extern void *lfind(const void *, const void *, size_t *, size_t,
458         __compar_fn_t);
459 extern void *lsearch(const void *, void *, size_t *, size_t,
460         __compar_fn_t);
461 extern void remque(void *);
462 extern void hdestroy(void);
463 extern void *tdelete(const void *, void **, __compar_fn_t);
464 extern void *tfind(const void *, void *const *, __compar_fn_t);
465 extern void *tsearch(const void *, void **, __compar_fn_t);
466 extern void twalk(const void *, __action_fn_t);

```

11.3.42 setjmp.h

```

467
468 typedef long int __jmp_buf[8];
469
470 extern int __sigsetjmp(jmp_buf, int);
471 extern void longjmp(jmp_buf, int);
472 extern void siglongjmp(sigjmp_buf, int);
473 extern void _longjmp(jmp_buf, int);
474 extern int _setjmp(jmp_buf);

```

11.3.43 signal.h

```

475
476 #define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int))-4)
477
478 #define SI_PAD_SIZE ((SI_MAX_SIZE/sizeof(int))-4)
479
480 struct sigaction {
481     union {
482         sighandler_t _sa_handler;
483         void (*_sa_sigaction) (int, siginfo_t *, void *);
484     } __sigaction_handler;
485     sigset_t sa_mask;
486     int sa_flags;
487     void (*sa_restorer) (void);
488 };
489
490 #define MINSIGSTKSZ 2048

```

11 Libraries

```
491         #define SIGSTKSZ             8192
492
493     struct _fpxreg {
494         unsigned short significand[4];
495         unsigned short exponent;
496         unsigned short padding[3];
497     };
498     struct _xmmreg {
499         uint32_t element[4];
500     };
501
502     struct _fpstate {
503         uint16_t cwd;
504         uint16_t swd;
505         uint16_t ftw;
506         uint16_t fop;
507         uint64_t rip;
508         uint64_t rdp;
509         uint32_t mxcsr;
510         uint32_t mxcr_mask;
511         struct _fpxreg _st[8];
512         struct _xmmreg _xmm[16];
513         uint32_t padding[24];
514     };
515
516     struct sigcontext {
517         unsigned long int r8;
518         unsigned long int r9;
519         unsigned long int r10;
520         unsigned long int r11;
521         unsigned long int r12;
522         unsigned long int r13;
523         unsigned long int r14;
524         unsigned long int r15;
525         unsigned long int rdi;
526         unsigned long int rsi;
527         unsigned long int rbp;
528         unsigned long int rbx;
529         unsigned long int rdx;
530         unsigned long int rax;
531         unsigned long int rcx;
532         unsigned long int rsp;
533         unsigned long int rip;
534         unsigned long int eflags;
535         unsigned short cs;
536         unsigned short gs;
537         unsigned short fs;
538         unsigned short __pad0;
539         unsigned long int err;
540         unsigned long int trapno;
541         unsigned long int oldmask;
542         unsigned long int cr2;
543         struct _fpstate *fpstate;
544         unsigned long int __reserved1[8];
545     };
546     extern int __libc_current_sigrtmax(void);
547     extern int __libc_current_sigrtmin(void);
548     extern sighandler_t __sysv_signal(int, sighandler_t);
549     extern char *const _sys_siglist(void);
550     extern int killpg(pid_t, int);
551     extern void psignal(int, const char *);
552     extern int raise(int);
553     extern int sigaddset(sigset_t *, int);
554     extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
```



```

555     extern int sigdelset(sigset_t *, int);
556     extern int sigemptyset(sigset_t *);
557     extern int sigfillset(sigset_t *);
558     extern int sighold(int);
559     extern int sigignore(int);
560     extern int siginterrupt(int, int);
561     extern int sigisemptyset(const sigset_t *);
562     extern int sigismember(const sigset_t *, int);
563     extern int sigorset(sigset_t *, const sigset_t *, const sigset_t *);
564     extern int sigpending(sigset_t *);
565     extern int sigrelse(int);
566     extern sighandler_t sigset(int, sighandler_t);
567     extern int pthread_kill(pthread_t, int);
568     extern int pthread_sigmask(int, sigset_t *, sigset_t *);
569     extern int sigaction(int, const struct sigaction *, struct sigaction *);
570     extern int sigwait(sigset_t *, int *);
571     extern int kill(pid_t, int);
572     extern int sigaltstack(const struct sigaltstack *, struct sigaltstack
573     *);
574     extern sighandler_t signal(int, sighandler_t);
575     extern int sigpause(int);
576     extern int sigprocmask(int, const sigset_t *, sigset_t *);
577     extern int sigreturn(struct sigcontext *);
578     extern int sigsuspend(const sigset_t *);
579     extern int sigqueue(pid_t, int, const union sigval);
580     extern int sigwaitinfo(const sigset_t *, siginfo_t *);
581     extern int sigtimedwait(const sigset_t *, siginfo_t *,
582     const struct timespec *);
583     extern sighandler_t bsd_signal(int, sighandler_t);

```

11.3.44 stddef.h

```

584
585     typedef long int ptrdiff_t;
586     typedef unsigned long int size_t;

```

11.3.45 stdio.h

```

587
588     #define __IO_FILE_SIZE 216
589
590     extern char *const _sys_errlist(void);
591     extern void clearerr(FILE *);
592     extern int fclose(FILE *);
593     extern FILE *fdopen(int, const char *);
594     extern int fflush_unlocked(FILE *);
595     extern int fileno(FILE *);
596     extern FILE *fopen(const char *, const char *);
597     extern int fprintf(FILE *, const char *, ...);
598     extern int fputc(int, FILE *);
599     extern FILE *freopen(const char *, const char *, FILE *);
600     extern FILE *freopen64(const char *, const char *, FILE *);
601     extern int fscanf(FILE *, const char *, ...);
602     extern int fseek(FILE *, long int, int);
603     extern int fseeko(FILE *, off_t, int);
604     extern int fseeko64(FILE *, loff_t, int);
605     extern off_t ftello(FILE *);
606     extern loff_t ftello64(FILE *);
607     extern int getchar(void);
608     extern int getchar_unlocked(void);
609     extern int getw(FILE *);
610     extern int pclose(FILE *);
611     extern void perror(const char *);
612     extern FILE *popen(const char *, const char *);

```

```

613     extern int printf(const char *, ...);
614     extern int putc_unlocked(int, FILE *);
615     extern int putchar(int);
616     extern int putchar_unlocked(int);
617     extern int putw(int, FILE *);
618     extern int remove(const char *);
619     extern void rewind(FILE *);
620     extern int scanf(const char *, ...);
621     extern void setbuf(FILE *, char *);
622     extern int sprintf(char *, const char *, ...);
623     extern int sscanf(const char *, const char *, ...);
624     extern FILE *stderr(void);
625     extern FILE *stdin(void);
626     extern FILE *stdout(void);
627     extern char *tempnam(const char *, const char *);
628     extern FILE *tmpfile64(void);
629     extern FILE *tmpfile(void);
630     extern char *tmpnam(char *);
631     extern int vfprintf(FILE *, const char *, va_list);
632     extern int vprintf(const char *, va_list);
633     extern int feof(FILE *);
634     extern int ferror(FILE *);
635     extern int fflush(FILE *);
636     extern int fgetc(FILE *);
637     extern int fgetpos(FILE *, fpos_t *);
638     extern char *fgets(char *, int, FILE *);
639     extern int fputs(const char *, FILE *);
640     extern size_t fread(void *, size_t, size_t, FILE *);
641     extern int fsetpos(FILE *, const fpos_t *);
642     extern long int ftell(FILE *);
643     extern size_t fwrite(const void *, size_t, size_t, FILE *);
644     extern int getc(FILE *);
645     extern int putc(int, FILE *);
646     extern int puts(const char *);
647     extern int setvbuf(FILE *, char *, int, size_t);
648     extern int snprintf(char *, size_t, const char *, ...);
649     extern int ungetc(int, FILE *);
650     extern int vsnprintf(char *, size_t, const char *, va_list);
651     extern int vsprintf(char *, const char *, va_list);
652     extern void flockfile(FILE *);
653     extern int asprintf(char **, const char *, ...);
654     extern int fgetpos64(FILE *, fpos64_t *);
655     extern FILE *fopen64(const char *, const char *);
656     extern int fsetpos64(FILE *, const fpos64_t *);
657     extern int ftrylockfile(FILE *);
658     extern void funlockfile(FILE *);
659     extern int getc_unlocked(FILE *);
660     extern void setbuffer(FILE *, char *, size_t);
661     extern int vasprintf(char **, const char *, va_list);
662     extern int vdprintf(int, const char *, va_list);
663     extern int vfscanf(FILE *, const char *, va_list);
664     extern int vscanf(const char *, va_list);
665     extern int vsscanf(const char *, const char *, va_list);
666     extern size_t __fpending(FILE *);

```

11.3.46 stdlib.h

```

667     extern double __strtod_internal(const char *, char **, int);
668     extern float __strtof_internal(const char *, char **, int);
669     extern long int __strtoul_internal(const char *, char **, int, int);
670     extern long double __strtold_internal(const char *, char **, int);
671     extern long long int __strtoll_internal(const char *, char **, int, int);
672     extern unsigned long int __strtoul_internal(const char *, char **, int,
673

```

```

674                                     int);
675     extern unsigned long long int __strtoull_internal(const char *, char **,
676                                                       int, int);
677     extern long int a64l(const char *);
678     extern void abort(void);
679     extern int abs(int);
680     extern double atof(const char *);
681     extern int atoi(char *);
682     extern long int atol(char *);
683     extern long long int atoll(const char *);
684     extern void *bsearch(const void *, const void *, size_t, size_t,
685                          __compar_fn_t);
686     extern div_t div(int, int);
687     extern double drand48(void);
688     extern char *ecvt(double, int, int *, int *);
689     extern double erand48(unsigned short);
690     extern void exit(int);
691     extern char *fcvt(double, int, int *, int *);
692     extern char *gcvt(double, int, char *);
693     extern char *getenv(const char *);
694     extern int getsuopt(char **, char *const *, char **);
695     extern int grantpt(int);
696     extern long int jrand48(unsigned short);
697     extern char *l64a(long int);
698     extern long int labs(long int);
699     extern void lcong48(unsigned short);
700     extern ldiv_t ldiv(long int, long int);
701     extern long long int llabs(long long int);
702     extern lldiv_t lldiv(long long int, long long int);
703     extern long int lrand48(void);
704     extern int mblen(const char *, size_t);
705     extern size_t mbstowcs(wchar_t *, const char *, size_t);
706     extern int mbtowc(wchar_t *, const char *, size_t);
707     extern char *mktemp(char *);
708     extern long int mrand48(void);
709     extern long int nrand48(unsigned short);
710     extern char *ptsname(int);
711     extern int putenv(char *);
712     extern void qsort(void *, size_t, size_t, __compar_fn_t);
713     extern int rand(void);
714     extern int rand_r(unsigned int *);
715     extern unsigned short *seed48(unsigned short);
716     extern void srand48(long int);
717     extern int unlockpt(int);
718     extern size_t wcstombs(char *, const wchar_t *, size_t);
719     extern int wctomb(char *, wchar_t);
720     extern int system(const char *);
721     extern void *calloc(size_t, size_t);
722     extern void free(void *);
723     extern char *initstate(unsigned int, char *, size_t);
724     extern void *malloc(size_t);
725     extern long int random(void);
726     extern void *realloc(void *, size_t);
727     extern char *setstate(char *);
728     extern void srand(unsigned int);
729     extern void srandom(unsigned int);
730     extern double strtod(char *, char **);
731     extern float strttof(const char *, char **);
732     extern long int strtol(char *, char **, int);
733     extern long double strtold(const char *, char **);
734     extern long long int strtoll(const char *, char **, int);
735     extern long long int strtoll(const char *, char **, int);
736     extern unsigned long int strtoul(const char *, char **, int);
737     extern unsigned long long int strtoull(const char *, char **, int);

```

```

738     extern unsigned long long int strtouq(const char *, char **, int);
739     extern void _Exit(int);
740     extern size_t __ctype_get_mb_cur_max(void);
741     extern char **environ(void);
742     extern char *realpath(const char *, char *);
743     extern int setenv(const char *, const char *, int);
744     extern int unsetenv(const char *);
745     extern int getloadavg(double, int);
746     extern int mkstemp64(char *);
747     extern int posix_memalign(void **, size_t, size_t);
748     extern int posix_openpt(int);

```

11.3.47 string.h

```

749
750     extern void *__memcpy(void *, const void *, size_t);
751     extern char *__stpcpy(char *, const char *);
752     extern char *__strtok_r(char *, const char *, char **);
753     extern void bcopy(void *, void *, size_t);
754     extern void *memchr(void *, int, size_t);
755     extern int memcmp(void *, void *, size_t);
756     extern void *memcpy(void *, void *, size_t);
757     extern void *memmem(const void *, size_t, const void *, size_t);
758     extern void *memmove(void *, const void *, size_t);
759     extern void *memset(void *, int, size_t);
760     extern char *strcat(char *, const char *);
761     extern char *strchr(char *, int);
762     extern int strcmp(char *, char *);
763     extern int strcoll(const char *, const char *);
764     extern char *strcpy(char *, char *);
765     extern size_t strcspn(const char *, const char *);
766     extern char *strerror(int);
767     extern size_t strlen(char *);
768     extern char *strncat(char *, char *, size_t);
769     extern int strncmp(char *, char *, size_t);
770     extern char *strncpy(char *, char *, size_t);
771     extern char *strpbrk(const char *, const char *);
772     extern char *strrchr(char *, int);
773     extern char *strsignal(int);
774     extern size_t strspn(const char *, const char *);
775     extern char *strstr(char *, char *);
776     extern char *strtok(char *, const char *);
777     extern size_t strxfrm(char *, const char *, size_t);
778     extern int bcmp(void *, void *, size_t);
779     extern void bzero(void *, size_t);
780     extern int ffs(int);
781     extern char *index(char *, int);
782     extern void *memccpy(void *, const void *, int, size_t);
783     extern char *rindex(char *, int);
784     extern int strcasecmp(char *, char *);
785     extern char *strdup(char *);
786     extern int strncasecmp(char *, char *, size_t);
787     extern char *strndup(const char *, size_t);
788     extern size_t strnlen(const char *, size_t);
789     extern char *strsep(char **, const char *);
790     extern char *strerror_r(int, char *, size_t);
791     extern char *strtok_r(char *, const char *, char **);
792     extern char *strcasestr(const char *, const char *);
793     extern char *stpcpy(char *, const char *);
794     extern char *stpncpy(char *, const char *, size_t);
795     extern void *memrchr(const void *, int, size_t);

```

11.3.48 sys/file.h

```
796
797 extern int flock(int, int);
```

11.3.49 sys/ioctl.h

```
798
799 #define TIOCGWINSZ      0x5413
800 #define FIONREAD        0x541B
801 #define TIOCNOTTY      21538
802
803 extern int ioctl(int, unsigned long int, ...);
```

11.3.50 sys/ipc.h

```
804
805 struct ipc_perm {
806     key_t __key;
807     uid_t uid;
808     gid_t gid;
809     uid_t cuid;
810     uid_t cgid;
811     unsigned short mode;
812     unsigned short __pad1;
813     unsigned short __seq;
814     unsigned short __pad2;
815     unsigned long int __unused1;
816     unsigned long int __unused2;
817 };
818
819 extern key_t ftok(char *, int);
```

11.3.51 sys/mman.h

```
820
821 #define MCL_CURRENT     1
822 #define MCL_FUTURE     2
823
824 extern int msync(void *, size_t, int);
825 extern int mlock(const void *, size_t);
826 extern int mlockall(int);
827 extern void *mmap(void *, size_t, int, int, int, off_t);
828 extern int mprotect(void *, size_t, int);
829 extern int munlock(const void *, size_t);
830 extern int munlockall(void);
831 extern int munmap(void *, size_t);
832 extern void *mmap64(void *, size_t, int, int, int, off64_t);
833 extern int shm_open(const char *, int, mode_t);
834 extern int shm_unlink(const char *);
```

11.3.52 sys/msg.h

```
835
836 typedef unsigned long int msgqnum_t;
837 typedef unsigned long int msglen_t;
838
839 struct msqid_ds {
840     struct ipc_perm msg_perm;
841     time_t msg_stime;
842     time_t msg_rtime;
843     time_t msg_ctime;
844     unsigned long int __msg_cbytes;
```

```

845         msgqnum_t msg_qnum;
846         msglen_t msg_qbytes;
847         pid_t msg_lspid;
848         pid_t msg_lrpid;
849         unsigned long int __unused4;
850         unsigned long int __unused5;
851     };
852     extern int msgctl(int, int, struct msqid_ds *);
853     extern int msgget(key_t, int);
854     extern int msgrcv(int, void *, size_t, long int, int);
855     extern int msgsnd(int, const void *, size_t, int);

```

11.3.53 sys/param.h

```

856
857     /*
858     * This header is architecture neutral
859     * Please refer to the generic specification for details
860     */

```

11.3.54 sys/poll.h

```

861
862     /*
863     * This header is architecture neutral
864     * Please refer to the generic specification for details
865     */

```

11.3.55 sys/resource.h

```

866
867     extern int getpriority(__priority_which_t, id_t);
868     extern int getrlimit64(id_t, struct rlimit64 *);
869     extern int setpriority(__priority_which_t, id_t, int);
870     extern int setrlimit(__rlimit_resource_t, const struct rlimit *);
871     extern int setrlimit64(__rlimit_resource_t, const struct rlimit64 *);
872     extern int getrlimit(__rlimit_resource_t, struct rlimit *);
873     extern int getrusage(int, struct rusage *);

```

11.3.56 sys/sem.h

```

874
875     struct semid_ds {
876         struct ipc_perm sem_perm;
877         time_t sem_otime;
878         unsigned long int __unused1;
879         time_t sem_ctime;
880         unsigned long int __unused2;
881         unsigned long int sem_nsems;
882         unsigned long int __unused3;
883         unsigned long int __unused4;
884     };
885     extern int semctl(int, int, int, ...);
886     extern int semget(key_t, int, int);
887     extern int semop(int, struct sembuf *, size_t);

```

11.3.57 sys/shm.h

```

888
889     #define SHMLBA (__getpagesize())
890
891     typedef unsigned long int shmatt_t;
892

```

```

893     struct shmids {
894         struct ipc_perm shm_perm;
895         size_t shm_segsz;
896         time_t shm_atime;
897         time_t shm_dtime;
898         time_t shm_ctime;
899         pid_t shm_cpid;
900         pid_t shm_lpid;
901         shmatt_t shm_nattch;
902         unsigned long int __unused4;
903         unsigned long int __unused5;
904     };
905     extern int __getpagesize(void);
906     extern void *shmat(int, const void *, int);
907     extern int shmctl(int, int, struct shmids *);
908     extern int shmdt(const void *);
909     extern int shmget(key_t, size_t, int);

```

11.3.58 sys/socket.h

```

910
911     typedef uint64_t __ss_aligntype;
912
913     #define SO_RCVLOWAT    18
914     #define SO_SNDLOWAT    19
915     #define SO_RCVTIMEO    20
916     #define SO_SNDTIMEO    21
917
918     extern int bind(int, const struct sockaddr *, socklen_t);
919     extern int getnameinfo(const struct sockaddr *, socklen_t, char *,
920                           socklen_t, char *, socklen_t, unsigned int);
921     extern int getsockname(int, struct sockaddr *, socklen_t *);
922     extern int listen(int, int);
923     extern int setsockopt(int, int, int, const void *, socklen_t);
924     extern int accept(int, struct sockaddr *, socklen_t *);
925     extern int connect(int, const struct sockaddr *, socklen_t);
926     extern ssize_t recv(int, void *, size_t, int);
927     extern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *,
928                             socklen_t *);
929     extern ssize_t recvmsg(int, struct msghdr *, int);
930     extern ssize_t send(int, const void *, size_t, int);
931     extern ssize_t sendmsg(int, const struct msghdr *, int);
932     extern ssize_t sendto(int, const void *, size_t, int,
933                          const struct sockaddr *, socklen_t);
934     extern int getpeername(int, struct sockaddr *, socklen_t *);
935     extern int getsockopt(int, int, int, void *, socklen_t *);
936     extern int shutdown(int, int);
937     extern int socket(int, int, int);
938     extern int socketpair(int, int, int, int);
939     extern int socketat(int);

```

11.3.59 sys/stat.h

```

940
941     #define _STAT_VER      1
942
943     struct stat {
944         dev_t st_dev;
945         ino_t st_ino;
946         nlink_t st_nlink;
947         mode_t st_mode;
948         uid_t st_uid;
949         gid_t st_gid;
950         int pad0;

```

```

951         dev_t st_rdev;
952         off_t st_size;
953         blksize_t st_blksize;
954         blkcnt_t st_blocks;
955         struct timespec st_atim;
956         struct timespec st_mtim;
957         struct timespec st_ctim;
958         unsigned long int __unused[3];
959     };
960     struct stat64 {
961         dev_t st_dev;
962         ino64_t st_ino;
963         nlink_t st_nlink;
964         mode_t st_mode;
965         uid_t st_uid;
966         gid_t st_gid;
967         int pad0;
968         dev_t st_rdev;
969         off_t st_size;
970         blksize_t st_blksize;
971         blkcnt64_t st_blocks;
972         struct timespec st_atim;
973         struct timespec st_mtim;
974         struct timespec st_ctim;
975         unsigned long int __unused[3];
976     };
977
978     extern int __fxstat(int, int, struct stat *);
979     extern int __fxstat64(int, int, struct stat64 *);
980     extern int __lxstat(int, char *, struct stat *);
981     extern int __lxstat64(int, const char *, struct stat64 *);
982     extern int __xmknod(int, const char *, mode_t, dev_t *);
983     extern int __xstat(int, const char *, struct stat *);
984     extern int __xstat64(int, const char *, struct stat64 *);
985     extern int mkfifo(const char *, mode_t);
986     extern int chmod(const char *, mode_t);
987     extern int fchmod(int, mode_t);
988     extern mode_t umask(mode_t);

```

11.3.60 sys/statvfs.h

```

989
990     struct statvfs64 {
991         unsigned long int f_bsize;
992         unsigned long int f_frsize;
993         fsblkcnt64_t f_blocks;
994         fsblkcnt64_t f_bfree;
995         fsblkcnt64_t f_bavail;
996         fsfilcnt64_t f_files;
997         fsfilcnt64_t f_ffree;
998         fsfilcnt64_t f_favail;
999         unsigned long int f_fsid;
1000        unsigned long int f_flag;
1001        unsigned long int f_namemax;
1002        int __f_spare[6];
1003    };
1004     struct statvfs {
1005         unsigned long int f_bsize;
1006         unsigned long int f_frsize;
1007         fsblkcnt_t f_blocks;
1008         fsblkcnt_t f_bfree;
1009         fsblkcnt_t f_bavail;
1010         fsfilcnt_t f_files;
1011         fsfilcnt_t f_ffree;

```



```

1012         fsfilcnt_t f_favail;
1013         unsigned long int f_fsid;
1014         unsigned long int f_flag;
1015         unsigned long int f_namemax;
1016         int __f_spare[6];
1017     };
1018     extern int fstatvfs(int, struct statvfs *);
1019     extern int fstatvfs64(int, struct statvfs64 *);
1020     extern int statvfs(const char *, struct statvfs *);
1021     extern int statvfs64(const char *, struct statvfs64 *);

```

11.3.61 sys/time.h

```

1022
1023     extern int getitimer(__itimer_which_t, struct itimerval *);
1024     extern int setitimer(__itimer_which_t, const struct itimerval *,
1025                         struct itimerval *);
1026     extern int adjtime(const struct timeval *, struct timeval *);
1027     extern int gettimeofday(struct timeval *, struct timezone *);
1028     extern int utimes(const char *, const struct timeval *);

```

11.3.62 sys/timeb.h

```

1029
1030     extern int ftime(struct timeb *);

```

11.3.63 sys/times.h

```

1031
1032     extern clock_t times(struct tms *);

```

11.3.64 sys/types.h

```

1033
1034     typedef long int int64_t;
1035
1036     typedef int64_t ssize_t;
1037
1038     #define __FDSET_LONGS    16

```

11.3.65 sys/uio.h

```

1039
1040     extern ssize_t readv(int, const struct iovec *, int);
1041     extern ssize_t writev(int, const struct iovec *, int);

```

11.3.66 sys/un.h

```

1042
1043     /*
1044     * This header is architecture neutral
1045     * Please refer to the generic specification for details
1046     */

```

11.3.67 sys/utsname.h

```

1047
1048     extern int uname(struct utsname *);

```

11.3.68 sys/wait.h

```

1049

```

```

1050     extern pid_t wait(int *);
1051     extern pid_t waitpid(pid_t, int *, int);
1052     extern pid_t wait4(pid_t, int *, int, struct rusage *);

```

11.3.69 syslog.h

```

1053
1054     extern void closelog(void);
1055     extern void openlog(const char *, int, int);
1056     extern int setlogmask(int);
1057     extern void syslog(int, const char *, ...);
1058     extern void vsyslog(int, const char *, va_list);

```

11.3.70 termios.h

```

1059
1060     #define OLCUC      0000002
1061     #define ONLCR     0000004
1062     #define XCASE     0000004
1063     #define NLDLY     0000400
1064     #define CR1       0001000
1065     #define IUCLC     0001000
1066     #define CR2       0002000
1067     #define CR3       0003000
1068     #define CRDLY     0003000
1069     #define TAB1      0004000
1070     #define TAB2      0010000
1071     #define TAB3      0014000
1072     #define TABDLY    0014000
1073     #define BS1       0020000
1074     #define BSDLY     0020000
1075     #define VT1       0040000
1076     #define VTDLY     0040000
1077     #define FF1       0100000
1078     #define FFDLY     0100000
1079
1080     #define VSUSP     10
1081     #define VEOL      11
1082     #define VREPRINT  12
1083     #define VDISCARD  13
1084     #define VWERASE   14
1085     #define VEOL2     16
1086     #define VMIN      6
1087     #define VSWTC     7
1088     #define VSTART    8
1089     #define VSTOP     9
1090
1091     #define IXON      0002000
1092     #define IXOFF     0010000
1093
1094     #define CS6       0000020
1095     #define CS7       0000040
1096     #define CS8       0000060
1097     #define CSIZE     0000060
1098     #define CSTOPB    0000100
1099     #define CREAD     0000200
1100     #define PARENB    0000400
1101     #define PARODD    0001000
1102     #define HUPCL     0002000
1103     #define CLOCAL    0004000
1104     #define VTIME     5
1105
1106     #define ISIG      0000001
1107     #define ICANON    0000002

```

```

1108     #define ECHOE    0000020
1109     #define ECHOK    0000040
1110     #define ECHONL   0000100
1111     #define NOFLSH   0000200
1112     #define TOSTOP   0000400
1113     #define ECHOCTL   0001000
1114     #define ECHOPRT  0002000
1115     #define ECHOKE   0004000
1116     #define FLUSHO   0010000
1117     #define PENDIN   0040000
1118     #define IEXTEN   0100000
1119
1120     extern speed_t cfgetispeed(const struct termios *);
1121     extern speed_t cfgetospeed(const struct termios *);
1122     extern void cfmakeraw(struct termios *);
1123     extern int cfsetispeed(struct termios *, speed_t);
1124     extern int cfsetospeed(struct termios *, speed_t);
1125     extern int cfsetspeed(struct termios *, speed_t);
1126     extern int tcflow(int, int);
1127     extern int tcflush(int, int);
1128     extern pid_t tcgetsid(int);
1129     extern int tcsendbreak(int, int);
1130     extern int tcsetattr(int, int, const struct termios *);
1131     extern int tcdrain(int);
1132     extern int tcgetattr(int, struct termios *);

```

11.3.71 time.h

```

1133
1134     extern int __daylight(void);
1135     extern long int __timezone(void);
1136     extern char *__tzname(void);
1137     extern char *asctime(const struct tm *);
1138     extern clock_t clock(void);
1139     extern char *ctime(const time_t *);
1140     extern char *ctime_r(const time_t *, char *);
1141     extern double difftime(time_t, time_t);
1142     extern struct tm *getdate(const char *);
1143     extern int getdate_err(void);
1144     extern struct tm *gmtime(const time_t *);
1145     extern struct tm *localtime(const time_t *);
1146     extern time_t mktime(struct tm *);
1147     extern int stime(const time_t *);
1148     extern size_t strftime(char *, size_t, const char *, const struct tm *);
1149     extern char *strptime(const char *, const char *, struct tm *);
1150     extern time_t time(time_t *);
1151     extern int nanosleep(const struct timespec *, struct timespec *);
1152     extern int daylight(void);
1153     extern long int timezone(void);
1154     extern char *tzname(void);
1155     extern void tzset(void);
1156     extern char *asctime_r(const struct tm *, char *);
1157     extern struct tm *gmtime_r(const time_t *, struct tm *);
1158     extern struct tm *localtime_r(const time_t *, struct tm *);
1159     extern int clock_getcpuclockid(pid_t, clockid_t *);
1160     extern int clock_getres(clockid_t, struct timespec *);
1161     extern int clock_gettime(clockid_t, struct timespec *);
1162     extern int clock_nanosleep(clockid_t, int, const struct timespec *,
1163                               struct timespec *);
1164     extern int clock_settime(clockid_t, const struct timespec *);
1165     extern int timer_create(clockid_t, struct sigevent *, timer_t *);
1166     extern int timer_delete(timer_t);
1167     extern int timer_getoverrun(timer_t);
1168     extern int timer_gettime(timer_t, struct itimerspec *);

```

```

1169     extern int timer_settime(timer_t, int, const struct itimerspec *,
1170                             struct itimerspec *);

```

11.3.72 ucontext.h

```

1171
1172     struct _libc_fpxreg {
1173         unsigned short significand[4];
1174         unsigned short exponent;
1175         unsigned short padding[3];
1176     };
1177
1178     typedef long int greg_t;
1179
1180     #define NGREG    23
1181
1182     typedef greg_t gregset_t[23];
1183
1184     struct _libc_xmmreg {
1185         uint32_t element[4];
1186     };
1187     struct _libc_fpstate {
1188         uint16_t cwd;
1189         uint16_t swd;
1190         uint16_t ftw;
1191         uint16_t fop;
1192         uint64_t rip;
1193         uint64_t rdp;
1194         uint32_t mxcsr;
1195         uint32_t mxcr_mask;
1196         struct _libc_fpxreg _st[8];
1197         struct _libc_xmmreg _xmm[16];
1198         uint32_t padding[24];
1199     };
1200     typedef struct _libc_fpstate *fpregset_t;
1201
1202     typedef struct {
1203         gregset_t gregs;
1204         fpregset_t fpregs;
1205         unsigned long int __reserved1[8];
1206     } mcontext_t;
1207
1208     typedef struct ucontext {
1209         unsigned long int uc_flags;
1210         struct ucontext *uc_link;
1211         stack_t uc_stack;
1212         mcontext_t uc_mcontext;
1213         sigset_t uc_sigmask;
1214         struct _libc_fpstate __fpregs_mem;
1215     } ucontext_t;
1216     extern int getcontext(ucontext_t *);
1217     extern int makecontext(ucontext_t *, void (*func) (void)
1218                          , int, ...);
1219     extern int setcontext(const struct ucontext *);
1220     extern int swapcontext(ucontext_t *, const struct ucontext *);

```

11.3.73 ulimit.h

```

1221
1222     extern long int ulimit(int, ...);

```

11.3.74 unistd.h

```

1223
1224     typedef long int intptr_t;
1225
1226     extern char **__environ(void);
1227     extern pid_t __getpgid(pid_t);
1228     extern void _exit(int);
1229     extern int acct(const char *);
1230     extern unsigned int alarm(unsigned int);
1231     extern int chown(const char *, uid_t, gid_t);
1232     extern int chroot(const char *);
1233     extern size_t confstr(int, char *, size_t);
1234     extern int creat(const char *, mode_t);
1235     extern int creat64(const char *, mode_t);
1236     extern char *ctermid(char *);
1237     extern char *cuserid(char *);
1238     extern int daemon(int, int);
1239     extern int execl(const char *, const char *, ...);
1240     extern int execlp(const char *, const char *, ...);
1241     extern int execlp(const char *, const char *, ...);
1242     extern int execv(const char *, char *const);
1243     extern int execvp(const char *, char *const);
1244     extern int fdatsync(int);
1245     extern int ftruncate64(int, off64_t);
1246     extern long int gethostid(void);
1247     extern char *getlogin(void);
1248     extern int getlogin_r(char *, size_t);
1249     extern int getopt(int, char *const, const char *);
1250     extern pid_t getpgrp(void);
1251     extern pid_t getsid(pid_t);
1252     extern char *getwd(char *);
1253     extern int lockf(int, int, off_t);
1254     extern int mkstemp(char *);
1255     extern int nice(int);
1256     extern char *optarg(void);
1257     extern int opterr(void);
1258     extern int optind(void);
1259     extern int optopt(void);
1260     extern int rename(const char *, const char *);
1261     extern int setegid(gid_t);
1262     extern int seteuid(uid_t);
1263     extern int sethostname(const char *, size_t);
1264     extern int setpgrp(void);
1265     extern void swab(const void *, void *, ssize_t);
1266     extern void sync(void);
1267     extern pid_t tcgetpgrp(int);
1268     extern int tcsetpgrp(int, pid_t);
1269     extern int truncate(const char *, off_t);
1270     extern int truncate64(const char *, off64_t);
1271     extern char *ttyname(int);
1272     extern unsigned int ualarm(useconds_t, useconds_t);
1273     extern int usleep(useconds_t);
1274     extern int close(int);
1275     extern int fsync(int);
1276     extern off_t lseek(int, off_t, int);
1277     extern int open(const char *, int, ...);
1278     extern int pause(void);
1279     extern ssize_t read(int, void *, size_t);
1280     extern ssize_t write(int, const void *, size_t);
1281     extern char *crypt(char *, char *);
1282     extern void encrypt(char *, int);
1283     extern void setkey(const char *);
1284     extern int access(const char *, int);

```

```

1285     extern int brk(void *);
1286     extern int chdir(const char *);
1287     extern int dup(int);
1288     extern int dup2(int, int);
1289     extern int execve(const char *, char *const, char *const);
1290     extern int fchdir(int);
1291     extern int fchown(int, uid_t, gid_t);
1292     extern pid_t fork(void);
1293     extern gid_t getegid(void);
1294     extern uid_t geteuid(void);
1295     extern gid_t getgid(void);
1296     extern int getgroups(int, gid_t);
1297     extern int gethostname(char *, size_t);
1298     extern pid_t getpgid(pid_t);
1299     extern pid_t getpid(void);
1300     extern uid_t getuid(void);
1301     extern int lchown(const char *, uid_t, gid_t);
1302     extern int link(const char *, const char *);
1303     extern int mkdir(const char *, mode_t);
1304     extern long int pathconf(const char *, int);
1305     extern int pipe(int);
1306     extern int readlink(const char *, char *, size_t);
1307     extern int rmdir(const char *);
1308     extern void *sbrk(ptrdiff_t);
1309     extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
1310     extern int setgid(gid_t);
1311     extern int setpgid(pid_t, pid_t);
1312     extern int setregid(gid_t, gid_t);
1313     extern int setreuid(uid_t, uid_t);
1314     extern pid_t setsid(void);
1315     extern int setuid(uid_t);
1316     extern unsigned int sleep(unsigned int);
1317     extern int symlink(const char *, const char *);
1318     extern long int sysconf(int);
1319     extern int unlink(const char *);
1320     extern pid_t vfork(void);
1321     extern ssize_t pread(int, void *, size_t, off_t);
1322     extern ssize_t pwrite(int, const void *, size_t, off_t);
1323     extern char **_environ(void);
1324     extern long int fpathconf(int, int);
1325     extern int ftruncate(int, off_t);
1326     extern char *getcwd(char *, size_t);
1327     extern int getpagesize(void);
1328     extern pid_t getppid(void);
1329     extern int isatty(int);
1330     extern loff_t lseek64(int, loff_t, int);
1331     extern int open64(const char *, int, ...);
1332     extern ssize_t pread64(int, void *, size_t, off64_t);
1333     extern ssize_t pwrite64(int, const void *, size_t, off64_t);
1334     extern int ttyname_r(int, char *, size_t);

```

11.3.75 utime.h

```

1335
1336     extern int utime(const char *, const struct utimbuf *);

```

11.3.76 utmp.h

```

1337
1338     struct lastlog {
1339         int32_t ll_time;
1340         char ll_line[UT_LINESIZE];
1341         char ll_host[UT_HOSTSIZE];
1342     };

```

```

1343
1344     struct utmp {
1345         short ut_type;
1346         pid_t ut_pid;
1347         char ut_line[UT_LINESIZE];
1348         char ut_id[4];
1349         char ut_user[UT_NAMESIZE];
1350         char ut_host[UT_HOSTSIZE];
1351         struct exit_status ut_exit;
1352         int ut_session;
1353         struct {
1354             int32_t tv_sec;
1355             int32_t tv_usec;
1356         } ut_tv;
1357         int32_t ut_addr_v6[4];
1358         char __unused[20];
1359     };
1360
1361     extern void endutent(void);
1362     extern struct utmp *getutent(void);
1363     extern void setutent(void);
1364     extern int getutent_r(struct utmp *, struct utmp **);
1365     extern int utmpname(const char *);
1366     extern int login_tty(int);
1367     extern void login(const struct utmp *);
1368     extern int logout(const char *);
1369     extern void logwtmp(const char *, const char *, const char *);

```

11.3.77 utmpx.h

```

1370
1371     struct utmpx {
1372         short ut_type;
1373         pid_t ut_pid;
1374         char ut_line[UT_LINESIZE];
1375         char ut_id[4];
1376         char ut_user[UT_NAMESIZE];
1377         char ut_host[UT_HOSTSIZE];
1378         struct exit_status ut_exit;
1379         int32_t ut_session;
1380         struct {
1381             int32_t tv_sec;
1382             int32_t tv_usec;
1383         } ut_tv;
1384         int32_t ut_addr_v6[4];
1385         char __unused[20];
1386     };
1387
1388     extern void endutxent(void);
1389     extern struct utmpx *getutxid(void);
1390     extern struct utmpx *getutxid(const struct utmpx *);
1391     extern struct utmpx *getutxline(const struct utmpx *);
1392     extern struct utmpx *pututxline(const struct utmpx *);
1393     extern void setutxent(void);

```

11.3.78 wchar.h

```

1394
1395     extern double __wcstod_internal(const wchar_t *, wchar_t **, int);
1396     extern float __wcstof_internal(const wchar_t *, wchar_t **, int);
1397     extern long int __wcstol_internal(const wchar_t *, wchar_t **, int,
1398     int);
1399     extern long double __wcstold_internal(const wchar_t *, wchar_t **, int);

```

```

1400     extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t *
1401     *,
1402     int, int);
1403     extern wchar_t *wscat(wchar_t *, const wchar_t *);
1404     extern wchar_t *wcschr(const wchar_t *, wchar_t);
1405     extern int wcscmp(const wchar_t *, const wchar_t *);
1406     extern int wscoll(const wchar_t *, const wchar_t *);
1407     extern wchar_t *wcscpy(wchar_t *, const wchar_t *);
1408     extern size_t wcsncpy(const wchar_t *, const wchar_t *);
1409     extern wchar_t *wcsdup(const wchar_t *);
1410     extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
1411     extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
1412     extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
1413     extern wchar_t *wcpbrk(const wchar_t *, const wchar_t *);
1414     extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
1415     extern size_t wcsspn(const wchar_t *, const wchar_t *);
1416     extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
1417     extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t * *);
1418     extern int wcswidth(const wchar_t *, size_t);
1419     extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
1420     extern int wctob(wint_t);
1421     extern int wcwidth(wchar_t);
1422     extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
1423     extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
1424     extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
1425     extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
1426     extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
1427     extern size_t mbrlen(const char *, size_t, mbstate_t *);
1428     extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
1429     extern int mbsinit(const mbstate_t *);
1430     extern size_t mbsnrtowcs(wchar_t *, const char **, size_t, size_t,
1431     mbstate_t *);
1432     extern size_t mbsrtowcs(wchar_t *, const char **, size_t, mbstate_t *);
1433     extern wchar_t *wcpncpy(wchar_t *, const wchar_t *);
1434     extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
1435     extern size_t wcrntomb(char *, wchar_t, mbstate_t *);
1436     extern size_t wcslen(const wchar_t *);
1437     extern size_t wcsnrtombs(char *, const wchar_t * *, size_t, size_t,
1438     mbstate_t *);
1439     extern size_t wcsrtombs(char *, const wchar_t * *, size_t, mbstate_t *);
1440     extern double wcstod(const wchar_t *, wchar_t * *);
1441     extern float wcstof(const wchar_t *, wchar_t * *);
1442     extern long int wcstol(const wchar_t *, wchar_t * *, int);
1443     extern long double wcstold(const wchar_t *, wchar_t * *);
1444     extern long long int wcstoll(const wchar_t *, wchar_t * *, int);
1445     extern unsigned long int wcstoul(const wchar_t *, wchar_t * *, int);
1446     extern unsigned long long int wcstoull(const wchar_t *, wchar_t * *, int);
1447     extern wchar_t *wswcs(const wchar_t *, const wchar_t *);
1448     extern int wscasecmp(const wchar_t *, const wchar_t *);
1449     extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
1450     extern size_t wcsnlen(const wchar_t *, size_t);
1451     extern long long int wcstoll(const wchar_t *, wchar_t * *, int);
1452     extern unsigned long long int wcstoull(const wchar_t *, wchar_t * *, int);
1453     extern wint_t btowc(int);
1454     extern wint_t fgetwc(FILE *);
1455     extern wint_t fgetwc_unlocked(FILE *);
1456     extern wchar_t *fgetws(wchar_t *, int, FILE *);
1457     extern wint_t fputwc(wchar_t, FILE *);
1458     extern int fputws(const wchar_t *, FILE *);
1459     extern int fwide(FILE *, int);
1460     extern int fwprintf(FILE *, const wchar_t *, ...);
1461     extern int fwscanf(FILE *, const wchar_t *, ...);
1462     extern wint_t getwc(FILE *);
1463     extern wint_t getwchar(void);

```



```

1464 extern wint_t putwc(wchar_t, FILE *);
1465 extern wint_t putwchar(wchar_t);
1466 extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
1467 extern int swscanf(const wchar_t *, const wchar_t *, ...);
1468 extern wint_t ungetwc(wint_t, FILE *);
1469 extern int vfwprintf(FILE *, const wchar_t *, va_list);
1470 extern int vwscanf(FILE *, const wchar_t *, va_list);
1471 extern int vswprintf(wchar_t *, size_t, const wchar_t *, va_list);
1472 extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
1473 extern int vwprintf(const wchar_t *, va_list);
1474 extern int vwscanf(const wchar_t *, va_list);
1475 extern size_t wcsftime(wchar_t *, size_t, const wchar_t *,
1476                       const struct tm *);
1477 extern int wprintf(const wchar_t *, ...);
1478 extern int wscanf(const wchar_t *, ...);

```

11.3.79 wctype.h

```

1479
1480 extern int iswblank(wint_t);
1481 extern wint_t towlower(wint_t);
1482 extern wint_t towupper(wint_t);
1483 extern wctrans_t wctrans(const char *);
1484 extern int iswalnum(wint_t);
1485 extern int iswalpha(wint_t);
1486 extern int iswcntrl(wint_t);
1487 extern int iswctype(wint_t, wctype_t);
1488 extern int iswdigit(wint_t);
1489 extern int iswgraph(wint_t);
1490 extern int iswlower(wint_t);
1491 extern int iswprint(wint_t);
1492 extern int iswpunct(wint_t);
1493 extern int iswspace(wint_t);
1494 extern int iswupper(wint_t);
1495 extern int iswxdigit(wint_t);
1496 extern wctype_t wctype(const char *);
1497 extern wint_t towctrans(wint_t, wctrans_t);

```

11.3.80 wordexp.h

```

1498
1499 extern int wordexp(const char *, wordexp_t *, int);
1500 extern void wordfree(wordexp_t *);

```

11.4 Interfaces for libm

1501 Table 11-24 defines the library name and shared object name for the libm library

1502 **Table 11-24 libm Definition**

Library:	libm
SONAME:	libm.so.6

1503

1504 The behavior of the interfaces in this library is specified by the following specifica-
1505 tions:

```

1506 [ISOC99] ISO C (1999)
    [LSB] This Specification
    [SUSv2] SUSv2
    [SUSv3] ISO POSIX (2003)

```

11.4.1 Math

1507

11.4.1.1 Interfaces for Math

1508

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

1509

1510

1511

Table 11-25 libm - Math Function Interfaces

<code>__finite</code> (GLIBC_2.2.5) [ISOC99]	<code>__finitef</code> (GLIBC_2.2.5) [ISOC99]	<code>__finitel</code> (GLIBC_2.2.5) [ISOC99]	<code>__fpclassify</code> (GLIBC_2.2.5) [LSB]
<code>__fpclassifyf</code> (GLIBC_2.2.5) [LSB]	<code>__fpclassifyf_l</code> (GLIBC_2.2.5) [ISOC99]	<code>__signbitl</code> (GLIBC_2.2.5) [ISOC99]	<code>acos</code> (GLIBC_2.2.5) [SUSv3]
<code>acosf</code> (GLIBC_2.2.5) [SUSv3]	<code>acosh</code> (GLIBC_2.2.5) [SUSv3]	<code>acoshf</code> (GLIBC_2.2.5) [SUSv3]	<code>acoshl</code> (GLIBC_2.2.5) [SUSv3]
<code>acosl</code> (GLIBC_2.2.5) [SUSv3]	<code>asin</code> (GLIBC_2.2.5) [SUSv3]	<code>asinf</code> (GLIBC_2.2.5) [SUSv3]	<code>asinh</code> (GLIBC_2.2.5) [SUSv3]
<code>asinhf</code> (GLIBC_2.2.5) [SUSv3]	<code>asinh_l</code> (GLIBC_2.2.5) [SUSv3]	<code>asinl</code> (GLIBC_2.2.5) [SUSv3]	<code>atan</code> (GLIBC_2.2.5) [SUSv3]
<code>atan2</code> (GLIBC_2.2.5) [SUSv3]	<code>atan2f</code> (GLIBC_2.2.5) [SUSv3]	<code>atan2l</code> (GLIBC_2.2.5) [SUSv3]	<code>atanf</code> (GLIBC_2.2.5) [SUSv3]
<code>atanh</code> (GLIBC_2.2.5) [SUSv3]	<code>atanhf</code> (GLIBC_2.2.5) [SUSv3]	<code>atanhl</code> (GLIBC_2.2.5) [SUSv3]	<code>atanl</code> (GLIBC_2.2.5) [SUSv3]
<code>cabs</code> (GLIBC_2.2.5) [SUSv3]	<code>cabsf</code> (GLIBC_2.2.5) [SUSv3]	<code>cabs_l</code> (GLIBC_2.2.5) [SUSv3]	<code>cacos</code> (GLIBC_2.2.5) [SUSv3]
<code>cacosf</code> (GLIBC_2.2.5) [SUSv3]	<code>cacosh</code> (GLIBC_2.2.5) [SUSv3]	<code>cacoshf</code> (GLIBC_2.2.5) [SUSv3]	<code>cacoshl</code> (GLIBC_2.2.5) [SUSv3]
<code>cacosl</code> (GLIBC_2.2.5) [SUSv3]	<code>carg</code> (GLIBC_2.2.5) [SUSv3]	<code>cargf</code> (GLIBC_2.2.5) [SUSv3]	<code>cargl</code> (GLIBC_2.2.5) [SUSv3]
<code>casin</code> (GLIBC_2.2.5) [SUSv3]	<code>casinf</code> (GLIBC_2.2.5) [SUSv3]	<code>casinh</code> (GLIBC_2.2.5) [SUSv3]	<code>casinhf</code> (GLIBC_2.2.5) [SUSv3]
<code>casinh_l</code> (GLIBC_2.2.5) [SUSv3]	<code>casinl</code> (GLIBC_2.2.5) [SUSv3]	<code>catan</code> (GLIBC_2.2.5) [SUSv3]	<code>catanf</code> (GLIBC_2.2.5) [SUSv3]
<code>catanh</code> (GLIBC_2.2.5) [SUSv3]	<code>catanhf</code> (GLIBC_2.2.5) [SUSv3]	<code>catanhl</code> (GLIBC_2.2.5) [SUSv3]	<code>catanl</code> (GLIBC_2.2.5) [SUSv3]
<code>cbrt</code> (GLIBC_2.2.5) [SUSv3]	<code>cbrtf</code> (GLIBC_2.2.5) [SUSv3]	<code>cbrtl</code> (GLIBC_2.2.5) [SUSv3]	<code>ccos</code> (GLIBC_2.2.5) [SUSv3]
<code>ccosf</code> (GLIBC_2.2.5) [SUSv3]	<code>ccosh</code> (GLIBC_2.2.5) [SUSv3]	<code>ccoshf</code> (GLIBC_2.2.5) [SUSv3]	<code>ccoshl</code> (GLIBC_2.2.5) [SUSv3]
<code>ccosl</code> (GLIBC_2.2.5) [SUSv3]	<code>ceil</code> (GLIBC_2.2.5) [SUSv3]	<code>ceilf</code> (GLIBC_2.2.5) [SUSv3]	<code>ceil_l</code> (GLIBC_2.2.5) [SUSv3]
<code>cexp</code> (GLIBC_2.2.5) [SUSv3]	<code>cexpf</code> (GLIBC_2.2.5) [SUSv3]	<code>cexpl</code> (GLIBC_2.2.5) [SUSv3]	<code>cimag</code> (GLIBC_2.2.5) [SUSv3]

cimagf(GLIBC_2.2.5) [SUSv3]	cimagl(GLIBC_2.2.5) [SUSv3]	clog(GLIBC_2.2.5) [SUSv3]	clog10(GLIBC_2.2.5) [ISOC99]
clog10f(GLIBC_2.2.5) [ISOC99]	clog10l(GLIBC_2.2.5) [ISOC99]	clogf(GLIBC_2.2.5) [SUSv3]	clogl(GLIBC_2.2.5) [SUSv3]
conj(GLIBC_2.2.5) [SUSv3]	conjf(GLIBC_2.2.5) [SUSv3]	conjl(GLIBC_2.2.5) [SUSv3]	copysign(GLIBC_2.2.5) [SUSv3]
copysignf(GLIBC_2.2.5) [SUSv3]	copysignl(GLIBC_2.2.5) [SUSv3]	cos(GLIBC_2.2.5) [SUSv3]	cosf(GLIBC_2.2.5) [SUSv3]
cosh(GLIBC_2.2.5) [SUSv3]	coshf(GLIBC_2.2.5) [SUSv3]	coshl(GLIBC_2.2.5) [SUSv3]	cosl(GLIBC_2.2.5) [SUSv3]
cpow(GLIBC_2.2.5) [SUSv3]	cpowf(GLIBC_2.2.5) [SUSv3]	cpowl(GLIBC_2.2.5) [SUSv3]	cproj(GLIBC_2.2.5) [SUSv3]
cprojf(GLIBC_2.2.5) [SUSv3]	cprojl(GLIBC_2.2.5) [SUSv3]	creal(GLIBC_2.2.5) [SUSv3]	crealf(GLIBC_2.2.5) [SUSv3]
creall(GLIBC_2.2.5) [SUSv3]	csin(GLIBC_2.2.5) [SUSv3]	csinf(GLIBC_2.2.5) [SUSv3]	csinh(GLIBC_2.2.5) [SUSv3]
csinhf(GLIBC_2.2.5) [SUSv3]	csinhl(GLIBC_2.2.5) [SUSv3]	csinl(GLIBC_2.2.5) [SUSv3]	csqrt(GLIBC_2.2.5) [SUSv3]
csqrtf(GLIBC_2.2.5) [SUSv3]	csqrtl(GLIBC_2.2.5) [SUSv3]	ctan(GLIBC_2.2.5) [SUSv3]	ctanf(GLIBC_2.2.5) [SUSv3]
ctanh(GLIBC_2.2.5) [SUSv3]	ctanhf(GLIBC_2.2.5) [SUSv3]	ctanhl(GLIBC_2.2.5) [SUSv3]	ctanl(GLIBC_2.2.5) [SUSv3]
dremf(GLIBC_2.2.5) [ISOC99]	dreml(GLIBC_2.2.5) [ISOC99]	erf(GLIBC_2.2.5) [SUSv3]	erfc(GLIBC_2.2.5) [SUSv3]
erfcf(GLIBC_2.2.5) [SUSv3]	erfcl(GLIBC_2.2.5) [SUSv3]	erff(GLIBC_2.2.5) [SUSv3]	erfl(GLIBC_2.2.5) [SUSv3]
exp(GLIBC_2.2.5) [SUSv3]	exp2(GLIBC_2.2.5) [SUSv3]	exp2f(GLIBC_2.2.5) [SUSv3]	exp2l(GLIBC_2.2.5) [SUSv3]
expf(GLIBC_2.2.5) [SUSv3]	expl(GLIBC_2.2.5) [SUSv3]	expm1(GLIBC_2.2.5) [SUSv3]	expm1f(GLIBC_2.2.5) [SUSv3]
expm1l(GLIBC_2.2.5) [SUSv3]	fabs(GLIBC_2.2.5) [SUSv3]	fabsf(GLIBC_2.2.5) [SUSv3]	fabsl(GLIBC_2.2.5) [SUSv3]
fdim(GLIBC_2.2.5) [SUSv3]	fdimf(GLIBC_2.2.5) [SUSv3]	fdiml(GLIBC_2.2.5) [SUSv3]	feclearexcept(GLIBC_2.2.5) [SUSv3]
fegetenv(GLIBC_2.2.5) [SUSv3]	fegetexceptflag(GLIBC_2.2.5) [SUSv3]	fegetround(GLIBC_2.2.5) [SUSv3]	feholdexcept(GLIBC_2.2.5) [SUSv3]
feraiseexcept(GLIBC_2.2.5) [SUSv3]	fesetenv(GLIBC_2.2.5) [SUSv3]	fesetexceptflag(GLIBC_2.2.5) [SUSv3]	fesetround(GLIBC_2.2.5) [SUSv3]
fetestexcept(GLIBC_2.2.5) [SUSv3]	feupdateenv(GLIBC_2.2.5) [SUSv3]	finite(GLIBC_2.2.5) [SUSv3]	finitef(GLIBC_2.2.5) [SUSv3]

C_2.2.5) [SUSv3]	BC_2.2.5) [SUSv3]) [SUSv2]	5) [ISOC99]
finitel(GLIBC_2.2.5) [ISOC99]	floor(GLIBC_2.2.5) [SUSv3]	floorf(GLIBC_2.2.5) [SUSv3]	floorl(GLIBC_2.2.5) [SUSv3]
fma(GLIBC_2.2.5) [SUSv3]	fmaf(GLIBC_2.2.5) [SUSv3]	fmal(GLIBC_2.2.5) [SUSv3]	fmax(GLIBC_2.2.5) [SUSv3]
fmaxf(GLIBC_2.2.5) [SUSv3]	fmaxl(GLIBC_2.2.5) [SUSv3]	fmin(GLIBC_2.2.5) [SUSv3]	fminf(GLIBC_2.2.5) [SUSv3]
fminl(GLIBC_2.2.5) [SUSv3]	fmod(GLIBC_2.2.5) [SUSv3]	fmodf(GLIBC_2.2.5) [SUSv3]	fmodl(GLIBC_2.2.5) [SUSv3]
frexp(GLIBC_2.2.5) [SUSv3]	frexpf(GLIBC_2.2.5) [SUSv3]	frexpl(GLIBC_2.2.5) [SUSv3]	gamma(GLIBC_2.2.5) [SUSv2]
gammaf(GLIBC_2.2.5) [ISOC99]	gammal(GLIBC_2.2.5) [ISOC99]	hypot(GLIBC_2.2.5) [SUSv3]	hypotf(GLIBC_2.2.5) [SUSv3]
hypotl(GLIBC_2.2.5) [SUSv3]	ilogb(GLIBC_2.2.5) [SUSv3]	ilogbf(GLIBC_2.2.5) [SUSv3]	ilogbl(GLIBC_2.2.5) [SUSv3]
j0(GLIBC_2.2.5) [SUSv3]	j0f(GLIBC_2.2.5) [ISOC99]	j0l(GLIBC_2.2.5) [ISOC99]	j1(GLIBC_2.2.5) [SUSv3]
j1f(GLIBC_2.2.5) [ISOC99]	j1l(GLIBC_2.2.5) [ISOC99]	jn(GLIBC_2.2.5) [SUSv3]	jnf(GLIBC_2.2.5) [ISOC99]
jnl(GLIBC_2.2.5) [ISOC99]	ldexp(GLIBC_2.2.5) [SUSv3]	ldexpf(GLIBC_2.2.5) [SUSv3]	ldexpl(GLIBC_2.2.5) [SUSv3]
lgamma(GLIBC_2.2.5) [SUSv3]	lgamma_r(GLIBC_2.2.5) [ISOC99]	lgammaf(GLIBC_2.2.5) [SUSv3]	lgammaf_r(GLIBC_2.2.5) [ISOC99]
lgammal(GLIBC_2.2.5) [SUSv3]	lgammal_r(GLIBC_2.2.5) [ISOC99]	llrint(GLIBC_2.2.5) [SUSv3]	llrintf(GLIBC_2.2.5) [SUSv3]
llrintl(GLIBC_2.2.5) [SUSv3]	llround(GLIBC_2.2.5) [SUSv3]	llroundf(GLIBC_2.2.5) [SUSv3]	llroundl(GLIBC_2.2.5) [SUSv3]
log(GLIBC_2.2.5) [SUSv3]	log10(GLIBC_2.2.5) [SUSv3]	log10f(GLIBC_2.2.5) [SUSv3]	log10l(GLIBC_2.2.5) [SUSv3]
log1p(GLIBC_2.2.5) [SUSv3]	log1pf(GLIBC_2.2.5) [SUSv3]	log1pl(GLIBC_2.2.5) [SUSv3]	log2(GLIBC_2.2.5) [SUSv3]
log2f(GLIBC_2.2.5) [SUSv3]	log2l(GLIBC_2.2.5) [SUSv3]	logb(GLIBC_2.2.5) [SUSv3]	logbf(GLIBC_2.2.5) [SUSv3]
logbl(GLIBC_2.2.5) [SUSv3]	logf(GLIBC_2.2.5) [SUSv3]	logl(GLIBC_2.2.5) [SUSv3]	lrint(GLIBC_2.2.5) [SUSv3]
lrintf(GLIBC_2.2.5) [SUSv3]	lrintl(GLIBC_2.2.5) [SUSv3]	lround(GLIBC_2.2.5) [SUSv3]	lroundf(GLIBC_2.2.5) [SUSv3]
lroundl(GLIBC_2.2.5) [SUSv3]	matherr(GLIBC_2.2.5) [ISOC99]	modf(GLIBC_2.2.5) [SUSv3]	modff(GLIBC_2.2.5) [SUSv3]
modfl(GLIBC_2.2.5) [SUSv3]	nan(GLIBC_2.2.5) [SUSv3]	nanf(GLIBC_2.2.5) [SUSv3]	nanl(GLIBC_2.2.5) [SUSv3]

<code>nearbyint(GLIBC_2.2.5) [SUSv3]</code>	<code>nearbyintf(GLIBC_2.2.5) [SUSv3]</code>	<code>nearbyintl(GLIBC_2.2.5) [SUSv3]</code>	<code>nextafter(GLIBC_2.2.5) [SUSv3]</code>
<code>nextafterf(GLIBC_2.2.5) [SUSv3]</code>	<code>nextafterl(GLIBC_2.2.5) [SUSv3]</code>	<code>nexttoward(GLIBC_2.2.5) [SUSv3]</code>	<code>nexttowardf(GLIBC_2.2.5) [SUSv3]</code>
<code>nexttowardl(GLIBC_2.2.5) [SUSv3]</code>	<code>pow(GLIBC_2.2.5) [SUSv3]</code>	<code>pow10(GLIBC_2.2.5) [ISOC99]</code>	<code>pow10f(GLIBC_2.2.5) [ISOC99]</code>
<code>pow10l(GLIBC_2.2.5) [ISOC99]</code>	<code>powf(GLIBC_2.2.5) [SUSv3]</code>	<code>powl(GLIBC_2.2.5) [SUSv3]</code>	<code>remainder(GLIBC_2.2.5) [SUSv3]</code>
<code>remainderf(GLIBC_2.2.5) [SUSv3]</code>	<code>remainderl(GLIBC_2.2.5) [SUSv3]</code>	<code>remquo(GLIBC_2.2.5) [SUSv3]</code>	<code>remquof(GLIBC_2.2.5) [SUSv3]</code>
<code>remquol(GLIBC_2.2.5) [SUSv3]</code>	<code>rint(GLIBC_2.2.5) [SUSv3]</code>	<code>rintf(GLIBC_2.2.5) [SUSv3]</code>	<code>rintl(GLIBC_2.2.5) [SUSv3]</code>
<code>round(GLIBC_2.2.5) [SUSv3]</code>	<code>roundf(GLIBC_2.2.5) [SUSv3]</code>	<code>roundl(GLIBC_2.2.5) [SUSv3]</code>	<code>scalb(GLIBC_2.2.5) [SUSv3]</code>
<code>scalbf(GLIBC_2.2.5) [ISOC99]</code>	<code>scalbl(GLIBC_2.2.5) [ISOC99]</code>	<code>scalbln(GLIBC_2.2.5) [SUSv3]</code>	<code>scalblnf(GLIBC_2.2.5) [SUSv3]</code>
<code>scalblnl(GLIBC_2.2.5) [SUSv3]</code>	<code>scalbn(GLIBC_2.2.5) [SUSv3]</code>	<code>scalbnf(GLIBC_2.2.5) [SUSv3]</code>	<code>scalbnl(GLIBC_2.2.5) [SUSv3]</code>
<code>significand(GLIBC_2.2.5) [ISOC99]</code>	<code>significandf(GLIBC_2.2.5) [ISOC99]</code>	<code>significandl(GLIBC_2.2.5) [ISOC99]</code>	<code>sin(GLIBC_2.2.5) [SUSv3]</code>
<code>sincos(GLIBC_2.2.5) [ISOC99]</code>	<code>sincosf(GLIBC_2.2.5) [ISOC99]</code>	<code>sincosl(GLIBC_2.2.5) [ISOC99]</code>	<code>sinf(GLIBC_2.2.5) [SUSv3]</code>
<code>sinh(GLIBC_2.2.5) [SUSv3]</code>	<code>sinhf(GLIBC_2.2.5) [SUSv3]</code>	<code>sinhl(GLIBC_2.2.5) [SUSv3]</code>	<code>sinl(GLIBC_2.2.5) [SUSv3]</code>
<code>sqrt(GLIBC_2.2.5) [SUSv3]</code>	<code>sqrtf(GLIBC_2.2.5) [SUSv3]</code>	<code>sqrtl(GLIBC_2.2.5) [SUSv3]</code>	<code>tan(GLIBC_2.2.5) [SUSv3]</code>
<code>tanf(GLIBC_2.2.5) [SUSv3]</code>	<code>tanh(GLIBC_2.2.5) [SUSv3]</code>	<code>tanhf(GLIBC_2.2.5) [SUSv3]</code>	<code>tanhf(GLIBC_2.2.5) [SUSv3]</code>
<code>tanl(GLIBC_2.2.5) [SUSv3]</code>	<code>tgamma(GLIBC_2.2.5) [SUSv3]</code>	<code>tgammaf(GLIBC_2.2.5) [SUSv3]</code>	<code>tgammaf(GLIBC_2.2.5) [SUSv3]</code>
<code>trunc(GLIBC_2.2.5) [SUSv3]</code>	<code>truncf(GLIBC_2.2.5) [SUSv3]</code>	<code>truncl(GLIBC_2.2.5) [SUSv3]</code>	<code>y0(GLIBC_2.2.5) [SUSv3]</code>
<code>y0f(GLIBC_2.2.5) [ISOC99]</code>	<code>y0l(GLIBC_2.2.5) [ISOC99]</code>	<code>y1(GLIBC_2.2.5) [SUSv3]</code>	<code>y1f(GLIBC_2.2.5) [ISOC99]</code>
<code>y1l(GLIBC_2.2.5) [ISOC99]</code>	<code>yn(GLIBC_2.2.5) [SUSv3]</code>	<code>ynf(GLIBC_2.2.5) [ISOC99]</code>	<code>ynl(GLIBC_2.2.5) [ISOC99]</code>

1512

1513

1514

1515

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

1516

Table 11-26 libm - Math Data Interfaces

1517

signgam(GLIBC_2.5) [SUSv3]			
----------------------------	--	--	--

11.5 Data Definitions for libm

1518

1519

1520

1521

1522

1523

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

1524

1525

1526

1527

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

1528

1529

1530

1531

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

11.5.1 complex.h

1532

1533

1534

1535

1536

1537

1538

1539

1540

1541

1542

1543

1544

1545

1546

1547

1548

1549

1550

1551

1552

1553

1554

1555

1556

1557

1558

1559

1560

1561

1562

1563

1564

1565

1566

```
extern double cabs(double complex);
extern float cabsf(float complex);
extern long double cabsl(long double complex);
extern double complex cacos(double complex);
extern float complex cacosf(float complex);
extern double complex cacosh(double complex);
extern float complex cacoshf(float complex);
extern long double complex cacoshl(long double complex);
extern long double complex cacosl(long double complex);
extern double carg(double complex);
extern float cargf(float complex);
extern long double cargl(long double complex);
extern double complex casin(double complex);
extern float complex casinf(float complex);
extern double complex casinh(double complex);
extern float complex casinhf(float complex);
extern long double complex casinhl(long double complex);
extern long double complex casinl(long double complex);
extern double complex catan(double complex);
extern float complex catanf(float complex);
extern double complex catanh(double complex);
extern float complex catanhf(float complex);
extern long double complex catanhl(long double complex);
extern long double complex catanl(long double complex);
extern double complex ccos(double complex);
extern float complex ccosf(float complex);
extern double complex ccosh(double complex);
extern float complex ccoshf(float complex);
extern long double complex ccoshl(long double complex);
extern long double complex ccosl(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexpl(long double complex);
extern double cimag(double complex);
```

```

1567     extern float cimagf(float complex);
1568     extern long double cimagl(long double complex);
1569     extern double complex clog(double complex);
1570     extern float complex clog10f(float complex);
1571     extern long double complex clog10l(long double complex);
1572     extern float complex clogf(float complex);
1573     extern long double complex clogl(long double complex);
1574     extern double complex conj(double complex);
1575     extern float complex conjf(float complex);
1576     extern long double complex conjl(long double complex);
1577     extern double complex cpow(double complex, double complex);
1578     extern float complex cpowf(float complex, float complex);
1579     extern long double complex cpowl(long double complex, long double
1580     complex);
1581     extern double complex cproj(double complex);
1582     extern float complex cprojf(float complex);
1583     extern long double complex cprojl(long double complex);
1584     extern double creal(double complex);
1585     extern float crealf(float complex);
1586     extern long double creall(long double complex);
1587     extern double complex csin(double complex);
1588     extern float complex csinf(float complex);
1589     extern double complex csinh(double complex);
1590     extern float complex csinhf(float complex);
1591     extern long double complex csinhl(long double complex);
1592     extern long double complex csinl(long double complex);
1593     extern double complex csqrt(double complex);
1594     extern float complex csqrtf(float complex);
1595     extern long double complex csqrtl(long double complex);
1596     extern double complex ctan(double complex);
1597     extern float complex ctanf(float complex);
1598     extern double complex ctanh(double complex);
1599     extern float complex ctanhf(float complex);
1600     extern long double complex ctanhl(long double complex);
1601     extern long double complex ctanl(long double complex);

```

11.5.2 fenv.h

```

1602
1603     #define FE_INVALID      0x01
1604     #define FE_DIVBYZERO   0x04
1605     #define FE_OVERFLOW    0x08
1606     #define FE_UNDERFLOW  0x10
1607     #define FE_INEXACT    0x20
1608
1609     #define FE_ALL_EXCEPT \
1610         (FE_INEXACT | FE_DIVBYZERO | FE_UNDERFLOW | FE_OVERFLOW |
1611         FE_INVALID)
1612
1613     #define FE_TONEAREST   0
1614     #define FE_DOWNWARD   0x400
1615     #define FE_UPWARD     0x800
1616     #define FE_TOWARDZERO 0xc00
1617
1618     typedef unsigned short fexcept_t;
1619
1620     typedef struct {
1621         unsigned short __control_word;
1622         unsigned short __unused1;
1623         unsigned short __status_word;
1624         unsigned short __unused2;
1625         unsigned short __tags;
1626         unsigned short __unused3;
1627         unsigned int __eip;

```

```

1628         unsigned short __cs_selector;
1629         unsigned int __opcode:11;
1630         unsigned int __unused4:5;
1631         unsigned int __data_offset;
1632         unsigned short __data_selector;
1633         unsigned short __unused5;
1634         unsigned int __mxcsr;
1635     } fenv_t;
1636
1637     #define FE_DFL_ENV          ((__const fenv_t *) -1)
1638
1639     extern int feclearexcept(int);
1640     extern int fegetenv(fenv_t *);
1641     extern int fegetexceptflag(fexcept_t *, int);
1642     extern int fegetround(void);
1643     extern int feholdexcept(fenv_t *);
1644     extern int feraiseexcept(int);
1645     extern int fesetenv(const fenv_t *);
1646     extern int fesetexceptflag(const fexcept_t *, int);
1647     extern int fesetround(int);
1648     extern int fetestexcept(int);
1649     extern int feupdateenv(const fenv_t *);

```

11.5.3 math.h

```

1650
1651     #define fpclassify(x)      \
1652         (sizeof (x) == sizeof (float) ? __fpclassifyf (x) : sizeof (x)
1653 == sizeof (double) ? __fpclassify (x) : __fpclassifyl (x))
1654     #define signbit(x)       \
1655         (sizeof (x) == sizeof (float)? __signbitf (x) : sizeof (x) ==
1656 sizeof (double)? __signbit (x) : __signbitl (x))
1657
1658     #define FP_ILOGB0        -2147483648
1659     #define FP_ILOGBNAN     -2147483648
1660
1661     extern int __finite(double);
1662     extern int __finitef(float);
1663     extern int __finitel(long double);
1664     extern int __isinf(double);
1665     extern int __isinff(float);
1666     extern int __isinfl(long double);
1667     extern int __isnan(double);
1668     extern int __isnanf(float);
1669     extern int __isnanl(long double);
1670     extern int __signbit(double);
1671     extern int __signbitf(float);
1672     extern int __fpclassify(double);
1673     extern int __fpclassifyf(float);
1674     extern int __fpclassifyl(long double);
1675     extern int signgam(void);
1676     extern double copysign(double, double);
1677     extern int finite(double);
1678     extern double frexp(double, int *);
1679     extern double ldexp(double, int);
1680     extern double modf(double, double *);
1681     extern double acos(double);
1682     extern double acosh(double);
1683     extern double asinh(double);
1684     extern double atanh(double);
1685     extern double asin(double);
1686     extern double atan(double);
1687     extern double atan2(double, double);
1688     extern double cbrt(double);

```



```

1689     extern double ceil(double);
1690     extern double cos(double);
1691     extern double cosh(double);
1692     extern double erf(double);
1693     extern double erfc(double);
1694     extern double exp(double);
1695     extern double expm1(double);
1696     extern double fabs(double);
1697     extern double floor(double);
1698     extern double fmod(double, double);
1699     extern double gamma(double);
1700     extern double hypot(double, double);
1701     extern int ilogb(double);
1702     extern double j0(double);
1703     extern double j1(double);
1704     extern double jn(int, double);
1705     extern double lgamma(double);
1706     extern double log(double);
1707     extern double log10(double);
1708     extern double loglp(double);
1709     extern double logb(double);
1710     extern double nextafter(double, double);
1711     extern double pow(double, double);
1712     extern double remainder(double, double);
1713     extern double rint(double);
1714     extern double scalb(double, double);
1715     extern double sin(double);
1716     extern double sinh(double);
1717     extern double sqrt(double);
1718     extern double tan(double);
1719     extern double tanh(double);
1720     extern double y0(double);
1721     extern double y1(double);
1722     extern double yn(int, double);
1723     extern float copysignf(float, float);
1724     extern long double copysignl(long double, long double);
1725     extern int finitef(float);
1726     extern int finitel(long double);
1727     extern float frexpf(float, int *);
1728     extern long double frexpl(long double, int *);
1729     extern float ldexpf(float, int);
1730     extern long double ldexpl(long double, int);
1731     extern float modff(float, float *);
1732     extern long double modfl(long double, long double *);
1733     extern double scalbln(double, long int);
1734     extern float scalblnf(float, long int);
1735     extern long double scalblnl(long double, long int);
1736     extern double scalbn(double, int);
1737     extern float scalbnf(float, int);
1738     extern long double scalbnl(long double, int);
1739     extern float acosf(float);
1740     extern float acoshf(float);
1741     extern long double acoshl(long double);
1742     extern long double acosl(long double);
1743     extern float asinf(float);
1744     extern float asinhf(float);
1745     extern long double asinhl(long double);
1746     extern long double asinl(long double);
1747     extern float atan2f(float, float);
1748     extern long double atan2l(long double, long double);
1749     extern float atanf(float);
1750     extern float atanhf(float);
1751     extern long double atanhhl(long double);
1752     extern long double atanl(long double);

```

```

1753     extern float cbrtf(float);
1754     extern long double cbrtl(long double);
1755     extern float ceilf(float);
1756     extern long double ceill(long double);
1757     extern float cosf(float);
1758     extern float coshf(float);
1759     extern long double coshl(long double);
1760     extern long double cosl(long double);
1761     extern float dremf(float, float);
1762     extern long double dreml(long double, long double);
1763     extern float erfcf(float);
1764     extern long double erfcl(long double);
1765     extern float erff(float);
1766     extern long double erfl(long double);
1767     extern double exp2(double);
1768     extern float exp2f(float);
1769     extern long double exp2l(long double);
1770     extern float expf(float);
1771     extern long double expl(long double);
1772     extern float expmlf(float);
1773     extern long double expmll(long double);
1774     extern float fabsf(float);
1775     extern long double fabsl(long double);
1776     extern double fdim(double, double);
1777     extern float fdimf(float, float);
1778     extern long double fdiml(long double, long double);
1779     extern float floorf(float);
1780     extern long double floorl(long double);
1781     extern double fma(double, double, double);
1782     extern float fmaf(float, float, float);
1783     extern long double fmal(long double, long double, long double);
1784     extern double fmax(double, double);
1785     extern float fmaxf(float, float);
1786     extern long double fmaxl(long double, long double);
1787     extern double fmin(double, double);
1788     extern float fminf(float, float);
1789     extern long double fminl(long double, long double);
1790     extern float fmodf(float, float);
1791     extern long double fmodl(long double, long double);
1792     extern float gammaf(float);
1793     extern long double gammal(long double);
1794     extern float hypotf(float, float);
1795     extern long double hypotl(long double, long double);
1796     extern int ilogbf(float);
1797     extern int ilogbl(long double);
1798     extern float j0f(float);
1799     extern long double j0l(long double);
1800     extern float j1f(float);
1801     extern long double j1l(long double);
1802     extern float jnf(int, float);
1803     extern long double jnl(int, long double);
1804     extern double lgamma_r(double, int *);
1805     extern float lgammaf(float);
1806     extern float lgammaf_r(float, int *);
1807     extern long double lgammal(long double);
1808     extern long double lgammal_r(long double, int *);
1809     extern long long int llrint(double);
1810     extern long long int llrintf(float);
1811     extern long long int llrintl(long double);
1812     extern long long int llround(double);
1813     extern long long int llroundf(float);
1814     extern long long int llroundl(long double);
1815     extern float log10f(float);
1816     extern long double log10l(long double);

```

```

1817     extern float log1pf(float);
1818     extern long double loglpl(long double);
1819     extern double log2(double);
1820     extern float log2f(float);
1821     extern long double log2l(long double);
1822     extern float logbf(float);
1823     extern long double logbl(long double);
1824     extern float logf(float);
1825     extern long double logl(long double);
1826     extern long int lrint(double);
1827     extern long int lrintf(float);
1828     extern long int lrintl(long double);
1829     extern long int lround(double);
1830     extern long int lroundf(float);
1831     extern long int lroundl(long double);
1832     extern int matherr(struct exception *);
1833     extern double nan(const char *);
1834     extern float nanf(const char *);
1835     extern long double nanl(const char *);
1836     extern double nearbyint(double);
1837     extern float nearbyintf(float);
1838     extern long double nearbyintl(long double);
1839     extern float nextafterf(float, float);
1840     extern long double nextafterl(long double, long double);
1841     extern double nexttoward(double, long double);
1842     extern float nexttowardf(float, long double);
1843     extern long double nexttowardl(long double, long double);
1844     extern double pow10(double);
1845     extern float pow10f(float);
1846     extern long double pow10l(long double);
1847     extern float powf(float, float);
1848     extern long double powl(long double, long double);
1849     extern float remainderf(float, float);
1850     extern long double remainderl(long double, long double);
1851     extern double remquo(double, double, int *);
1852     extern float remquof(float, float, int *);
1853     extern long double remquol(long double, long double, int *);
1854     extern float rintf(float);
1855     extern long double rintl(long double);
1856     extern double round(double);
1857     extern float roundf(float);
1858     extern long double roundl(long double);
1859     extern float scalbf(float, float);
1860     extern long double scalbl(long double, long double);
1861     extern double significand(double);
1862     extern float significandf(float);
1863     extern long double significandl(long double);
1864     extern void sincos(double, double *, double *);
1865     extern void sincosf(float, float *, float *);
1866     extern void sincosl(long double, long double *, long double *);
1867     extern float sinf(float);
1868     extern float sinhf(float);
1869     extern long double sinhl(long double);
1870     extern long double sinl(long double);
1871     extern float sqrtf(float);
1872     extern long double sqrtl(long double);
1873     extern float tanf(float);
1874     extern float tanhf(float);
1875     extern long double tanhl(long double);
1876     extern long double tanl(long double);
1877     extern double tgamma(double);
1878     extern float tgammaf(float);
1879     extern long double tgamma_l(long double);
1880     extern double trunc(double);

```

```

1881     extern float truncf(float);
1882     extern long double trunc1(long double);
1883     extern float y0f(float);
1884     extern long double y0l(long double);
1885     extern float y1f(float);
1886     extern long double y1l(long double);
1887     extern float ynf(int, float);
1888     extern long double ynl(int, long double);
1889     extern int __fpclassify1(long double);
1890     extern int __fpclassify1(long double);
1891     extern int __signbit1(long double);
1892     extern int __signbit1(long double);
1893     extern int __signbit1(long double);
1894     extern long double exp2l(long double);
1895     extern long double exp2l(long double);

```

11.6 Interfaces for libpthread

1896 Table 11-27 defines the library name and shared object name for the libpthread
 1897 library

1898 **Table 11-27 libpthread Definition**

Library:	libpthread
SONAME:	libpthread.so.0

1899
 1900 The behavior of the interfaces in this library is specified by the following specifica-
 1901 tions:

- [LFS] Large File Support
- [LSB] This Specification
- [SUSv3] ISO POSIX (2003)

11.6.1 Realtime Threads

11.6.1.1 Interfaces for Realtime Threads

1903
 1904 An LSB conforming implementation shall provide the architecture specific functions
 1905 for Realtime Threads specified in Table 11-28, with the full mandatory functionality
 1906 as described in the referenced underlying specification.

1907 **Table 11-28 libpthread - Realtime Threads Function Interfaces**

pthread_attr_getinheritched(GLIB C_2.2.5) [SUSv3]	pthread_attr_getschedpolicy(GLIB C_2.2.5) [SUSv3]	pthread_attr_getschedpolicy(GLIBC_2.2.5) [SUSv3]	pthread_attr_setinheritched(GLIBC_2.2.5) [SUSv3]
pthread_attr_setschedpolicy(GLIBC_2.2.5) [SUSv3]	pthread_attr_setschedpolicy(GLIBC_2.2.5) [SUSv3]	pthread_getschedparam(GLIBC_2.2.5) [SUSv3]	pthread_setschedparam(GLIBC_2.2.5) [SUSv3]

11.6.2 Advanced Realtime Threads

11.6.2.1 Interfaces for Advanced Realtime Threads

1909
 1910 No external functions are defined for libpthread - Advanced Realtime Threads in
 1911 this part of the specification. See also the generic specification.

11.6.3 Posix Threads

11.6.3.1 Interfaces for Posix Threads

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

Table 11-29 libpthread - Posix Threads Function Interfaces

_pthread_cleanup_pop(GLIBC_2.2.5) [LSB]	_pthread_cleanup_push(GLIBC_2.2.5) [LSB]	pthread_attr_destroy(GLIBC_2.2.5) [SUSv3]	pthread_attr_getdetachstate(GLIBC_2.2.5) [SUSv3]
pthread_attr_getguardsize(GLIBC_2.2.5) [SUSv3]	pthread_attr_getschedparam(GLIBC_2.2.5) [SUSv3]	pthread_attr_getstack(GLIBC_2.2.5) [SUSv3]	pthread_attr_getstackaddr(GLIBC_2.2.5) [SUSv3]
pthread_attr_getstacksize(GLIBC_2.2.5) [SUSv3]	pthread_attr_init(GLIBC_2.2.5) [SUSv3]	pthread_attr_setdetachstate(GLIBC_2.2.5) [SUSv3]	pthread_attr_setguardsize(GLIBC_2.2.5) [SUSv3]
pthread_attr_setschedparam(GLIBC_2.2.5) [SUSv3]	pthread_attr_setstackaddr(GLIBC_2.2.5) [SUSv3]	pthread_attr_setstacksize(GLIBC_2.2.5) [SUSv3]	pthread_cancel(GLIBC_2.2.5) [SUSv3]
pthread_cond_broadcast(GLIBC_2.3.2) [SUSv3]	pthread_cond_destroy(GLIBC_2.3.2) [SUSv3]	pthread_cond_init(GLIBC_2.3.2) [SUSv3]	pthread_cond_signal(GLIBC_2.3.2) [SUSv3]
pthread_cond_timedwait(GLIBC_2.3.2) [SUSv3]	pthread_cond_wait(GLIBC_2.3.2) [SUSv3]	pthread_condattr_destroy(GLIBC_2.2.5) [SUSv3]	pthread_condattr_getshared(GLIBC_2.2.5) [SUSv3]
pthread_condattr_init(GLIBC_2.2.5) [SUSv3]	pthread_condattr_setshared(GLIBC_2.2.5) [SUSv3]	pthread_create(GLIBC_2.2.5) [SUSv3]	pthread_detach(GLIBC_2.2.5) [SUSv3]
pthread_equal(GLIBC_2.2.5) [SUSv3]	pthread_exit(GLIBC_2.2.5) [SUSv3]	pthread_getconcurrency(GLIBC_2.2.5) [SUSv3]	pthread_getspecific(GLIBC_2.2.5) [SUSv3]
pthread_join(GLIBC_2.2.5) [SUSv3]	pthread_key_create(GLIBC_2.2.5) [SUSv3]	pthread_key_delete(GLIBC_2.2.5) [SUSv3]	pthread_kill(GLIBC_2.2.5) [SUSv3]
pthread_mutex_destroy(GLIBC_2.2.5) [SUSv3]	pthread_mutex_init(GLIBC_2.2.5) [SUSv3]	pthread_mutex_lock(GLIBC_2.2.5) [SUSv3]	pthread_mutex_trylock(GLIBC_2.2.5) [SUSv3]
pthread_mutex_unlock(GLIBC_2.2.5) [SUSv3]	pthread_mutexattr_destroy(GLIBC_2.2.5) [SUSv3]	pthread_mutexattr_getshared(GLIBC_2.2.5) [SUSv3]	pthread_mutexattr_gettype(GLIBC_2.2.5) [SUSv3]
pthread_mutexattr_init(GLIBC_2.2.5) [SUSv3]	pthread_mutexattr_setshared(GLIBC_2.2.5) [SUSv3]	pthread_mutexattr_settype(GLIBC_2.2.5) [SUSv3]	pthread_once(GLIBC_2.2.5) [SUSv3]
pthread_rwlock_d	pthread_rwlock_i	pthread_rwlock_r	pthread_rwlock_ti

estroy(GLIBC_2.5) [SUSv3]	nit(GLIBC_2.2.5) [SUSv3]	dlock(GLIBC_2.2.5) [SUSv3]	medrdlock(GLIBC_2.2.5) [SUSv3]
pthread_rwlock_timedwait(GLIBC_2.2.5) [SUSv3]	pthread_rwlock_timedwait(GLIBC_2.2.5) [SUSv3]	pthread_rwlock_timedwait(GLIBC_2.2.5) [SUSv3]	pthread_rwlock_unlock(GLIBC_2.2.5) [SUSv3]
pthread_rwlock_wrlock(GLIBC_2.2.5) [SUSv3]	pthread_rwlockat_destroy(GLIBC_2.2.5) [SUSv3]	pthread_rwlockat_getpshared(GLIBC_2.2.5) [SUSv3]	pthread_rwlockat_init(GLIBC_2.2.5) [SUSv3]
pthread_rwlockat_setpshared(GLIBC_2.2.5) [SUSv3]	pthread_self(GLIBC_2.2.5) [SUSv3]	pthread_setcancelstate(GLIBC_2.2.5) [SUSv3]	pthread_setcanceltype(GLIBC_2.2.5) [SUSv3]
pthread_setconcurrency(GLIBC_2.2.5) [SUSv3]	pthread_setspecific(GLIBC_2.2.5) [SUSv3]	pthread_sigmask(GLIBC_2.2.5) [SUSv3]	pthread_testcancel(GLIBC_2.2.5) [SUSv3]
sem_close(GLIBC_2.2.5) [SUSv3]	sem_destroy(GLIBC_2.2.5) [SUSv3]	sem_getvalue(GLIBC_2.2.5) [SUSv3]	sem_init(GLIBC_2.2.5) [SUSv3]
sem_open(GLIBC_2.2.5) [SUSv3]	sem_post(GLIBC_2.2.5) [SUSv3]	sem_timedwait(GLIBC_2.2.5) [SUSv3]	sem_trywait(GLIBC_2.2.5) [SUSv3]
sem_unlink(GLIBC_2.2.5) [SUSv3]	sem_wait(GLIBC_2.2.5) [SUSv3]		

1917

11.6.4 Thread aware versions of libc interfaces

1918

11.6.4.1 Interfaces for Thread aware versions of libc interfaces

1919

An LSB conforming implementation shall provide the architecture specific functions for Thread aware versions of libc interfaces specified in Table 11-30, with the full mandatory functionality as described in the referenced underlying specification.

1920

1921

1922

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

1923

Interfaces

lseek64(GLIBC_2.2.5) [LFS]	open64(GLIBC_2.2.5) [LFS]	pread(GLIBC_2.2.5) [SUSv3]	pread64(GLIBC_2.2.5) [LFS]
pwrite(GLIBC_2.2.5) [SUSv3]	pwrite64(GLIBC_2.2.5) [LFS]		

1924

11.7 Data Definitions for libpthread

1925

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

1926

1927

1928

1929

1930

1931

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and

1932

1933 application developers should use this ABI to supplement - not to replace - source
1934 interface definition specifications.

1935 This specification uses the ISO C (1999) C Language as the reference programming
1936 language, and data definitions are specified in ISO C format. The C language is used
1937 here as a convenient notation. Using a C language description of these data objects
1938 does not preclude their use by other programming languages.

11.7.1 pthread.h

```

1939
1940 extern void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *,
1941 int);
1942 extern void _pthread_cleanup_push(struct _pthread_cleanup_buffer *,
1943 void (*__routine) (void *)
1944 , void *);
1945 extern int pthread_attr_destroy(pthread_attr_t *);
1946 extern int pthread_attr_getdetachstate(const typedef struct {
1947 int __detachstate;
1948 int __schedpolicy;
1949 struct sched_param
1950 __schedparam;
1951 int __inheritsched;
1952 int __scope;
1953 size_t __guardsize;
1954 int __stackaddr_set;
1955 void *__stackaddr;
1956 unsigned long int __stacksize;}
1957 pthread_attr_t *, int *);
1958 extern int pthread_attr_getinheritsched(const typedef struct {
1959 int __detachstate;
1960 int __schedpolicy;
1961 struct sched_param
1962 __schedparam;
1963 int __inheritsched;
1964 int __scope;
1965 size_t __guardsize;
1966 int __stackaddr_set;
1967 void *__stackaddr;
1968 unsigned long int
1969 __stacksize;}
1970 pthread_attr_t *, int *);
1971 extern int pthread_attr_getschedparam(const typedef struct {
1972 int __detachstate;
1973 int __schedpolicy;
1974 struct sched_param
1975 __schedparam;
1976 int __inheritsched;
1977 int __scope;
1978 size_t __guardsize;
1979 int __stackaddr_set;
1980 void *__stackaddr;
1981 unsigned long int __stacksize;}
1982 pthread_attr_t *, struct
1983 sched_param {
1984 int sched_priority;}
1985
1986 *);
1987 extern int pthread_attr_getschedpolicy(const typedef struct {
1988 int __detachstate;
1989 int __schedpolicy;
1990 struct sched_param
1991 __schedparam;

```

```

1992                                     int __inheritsched;
1993                                     int __scope;
1994                                     size_t __guardsize;
1995                                     int __stackaddr_set;
1996                                     void *__stackaddr;
1997                                     unsigned long int __stacksize;}
1998                                     pthread_attr_t *, int *);
1999 extern int pthread_attr_getscope(const typedef struct {
2000                                     int __detachstate;
2001                                     int __schedpolicy;
2002                                     struct sched_param __schedparam;
2003                                     int __inheritsched;
2004                                     int __scope;
2005                                     size_t __guardsize;
2006                                     int __stackaddr_set;
2007                                     void *__stackaddr;
2008                                     unsigned long int __stacksize;}
2009                                     pthread_attr_t *, int *);
2010 extern int pthread_attr_init(pthread_attr_t *);
2011 extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
2012 extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
2013 extern int pthread_attr_setschedparam(pthread_attr_t *, const struct
2014 sched_param {
2015                                     int sched_priority;}
2016                                     *);
2017                                     *);
2018 extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
2019 extern int pthread_attr_setscope(pthread_attr_t *, int);
2020 extern int pthread_cancel(typedef unsigned long int pthread_t);
2021 extern int pthread_cond_broadcast(pthread_cond_t *);
2022 extern int pthread_cond_destroy(pthread_cond_t *);
2023 extern int pthread_cond_init(pthread_cond_t *, const typedef struct {
2024                                     int __dummy;}
2025                                     pthread_condattr_t *);
2026                                     pthread_condattr_t *);
2027 extern int pthread_cond_signal(pthread_cond_t *);
2028 extern int pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
2029 const struct timespec {
2030                                     time_t tv_sec; long int tv_nsec;}
2031                                     *);
2032                                     *);
2033 extern int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
2034 extern int pthread_condattr_destroy(pthread_condattr_t *);
2035 extern int pthread_condattr_init(pthread_condattr_t *);
2036 extern int pthread_create(pthread_t *, const typedef struct {
2037                                     int __detachstate;
2038                                     int __schedpolicy;
2039                                     struct sched_param __schedparam;
2040                                     int __inheritsched;
2041                                     int __scope;
2042                                     size_t __guardsize;
2043                                     int __stackaddr_set;
2044                                     void *__stackaddr;
2045                                     unsigned long int __stacksize;}
2046                                     pthread_attr_t *,
2047                                     void *(*__start_routine) (void *p1)
2048                                     , void *);
2049 extern int pthread_detach(typedef unsigned long int pthread_t);
2050 extern int pthread_equal(typedef unsigned long int pthread_t,
2051                          typedef unsigned long int pthread_t);
2052 extern void pthread_exit(void *);
2053 extern int pthread_getschedparam(typedef unsigned long int pthread_t,
2054 int *, struct sched_param {
2055                                     int sched_priority;}

```



```

2056
2057
2058 extern void *pthread_getspecific(typedef unsigned int pthread_key_t);
2059 extern int pthread_join(typedef unsigned long int pthread_t, void **);
2060 extern int pthread_key_create(pthread_key_t *, void (*destr_func) (void
2061 *)
2062 );
2063 extern int pthread_key_delete(typedef unsigned int pthread_key_t);
2064 extern int pthread_mutex_destroy(pthread_mutex_t *);
2065 extern int pthread_mutex_init(pthread_mutex_t *, const typedef struct
2066 {
2067     int __mutexkind;}
2068
2069     pthread_mutexattr_t *);
2070 extern int pthread_mutex_lock(pthread_mutex_t *);
2071 extern int pthread_mutex_trylock(pthread_mutex_t *);
2072 extern int pthread_mutex_unlock(pthread_mutex_t *);
2073 extern int pthread_mutexattr_destroy(pthread_mutexattr_t *);
2074 extern int pthread_mutexattr_init(pthread_mutexattr_t *);
2075 extern int pthread_once(pthread_once_t *, void (*init_routine) (void)
2076 );
2077 extern int pthread_rwlock_destroy(pthread_rwlock_t *);
2078 extern int pthread_rwlock_init(pthread_rwlock_t *,
2079 pthread_rwlockattr_t *);
2080 extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
2081 extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *);
2082 extern int pthread_rwlock_trywrlock(pthread_rwlock_t *);
2083 extern int pthread_rwlock_unlock(pthread_rwlock_t *);
2084 extern int pthread_rwlock_wrlock(pthread_rwlock_t *);
2085 extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
2086 extern int pthread_rwlockattr_getpshared(const typedef struct {
2087     int __lockkind; int
2088     __pshared;}
2089     pthread_rwlockattr_t *, int
2090 );
2091 extern int pthread_rwlockattr_init(pthread_rwlockattr_t *);
2092 extern int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
2093 extern typedef unsigned long int pthread_t pthread_self(void);
2094 extern int pthread_setcancelstate(int, int *);
2095 extern int pthread_setcanceltype(int, int *);
2096 extern int pthread_setschedparam(typedef unsigned long int pthread_t,
2097 int, const struct sched_param {
2098     int sched_priority;}
2099
2100 );
2101 extern int pthread_setspecific(typedef unsigned int pthread_key_t,
2102 const void *);
2103 extern void pthread_testcancel(void);
2104 extern int pthread_attr_getguardsize(const typedef struct {
2105     int __detachstate;
2106     int __schedpolicy;
2107     struct sched_param __schedparam;
2108     int __inheritsched;
2109     int __scope;
2110     size_t __guardsize;
2111     int __stackaddr_set;
2112     void *__stackaddr;
2113     unsigned long int __stacksize;}
2114     pthread_attr_t *, size_t *);
2115 extern int pthread_attr_setguardsize(pthread_attr_t *,
2116 typedef unsigned long int
2117 size_t);
2118 extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
2119 extern int pthread_attr_getstackaddr(const typedef struct {

```

```

2120         int __detachstate;
2121         int __schedpolicy;
2122         struct sched_param __schedparam;
2123         int __inheritsched;
2124         int __scope;
2125         size_t __guardsize;
2126         int __stackaddr_set;
2127         void *__stackaddr;
2128         unsigned long int __stacksize;
2129         pthread_attr_t *, void **);
2130 extern int pthread_attr_setstacksize(pthread_attr_t *,
2131                                     typedef unsigned long int
2132                                     size_t);
2133 extern int pthread_attr_getstacksize(const typedef struct {
2134         int __detachstate;
2135         int __schedpolicy;
2136         struct sched_param __schedparam;
2137         int __inheritsched;
2138         int __scope;
2139         size_t __guardsize;
2140         int __stackaddr_set;
2141         void *__stackaddr;
2142         unsigned long int __stacksize;
2143         pthread_attr_t *, size_t *);
2144 extern int pthread_mutexattr_gettype(const typedef struct {
2145         int __mutexkind;
2146         pthread_mutexattr_t *, int *);
2147 extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
2148 extern int pthread_getconcurrency(void);
2149 extern int pthread_setconcurrency(int);
2150 extern int pthread_attr_getstack(const typedef struct {
2151         int __detachstate;
2152         int __schedpolicy;
2153         struct sched_param __schedparam;
2154         int __inheritsched;
2155         int __scope;
2156         size_t __guardsize;
2157         int __stackaddr_set;
2158         void *__stackaddr;
2159         unsigned long int __stacksize;
2160         pthread_attr_t *, void **, size_t *);
2161 extern int pthread_attr_setstack(pthread_attr_t *, void *,
2162                                 typedef unsigned long int size_t);
2163 extern int pthread_condattr_getpshared(const typedef struct {
2164         int __dummy;
2165         pthread_condattr_t *, int *);
2166 extern int pthread_condattr_setpshared(pthread_condattr_t *, int);
2167 extern int pthread_mutexattr_getpshared(const typedef struct {
2168         int __mutexkind;
2169         pthread_mutexattr_t *, int *);
2170 extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
2171 extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *, const struct
2172 timespec {
2173         time_t tv_sec; long int
2174         tv_nsec;
2175
2176         *);
2177 extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *, const struct
2178 timespec {
2179         time_t tv_sec; long int
2180         tv_nsec;
2181
2182         *);
2183 extern int __register_atfork(void (*prepare) (void)

```

```

2184         , void (*parent) (void)
2185         , void (*child) (void)
2186         , void *);
2187 extern int pthread_setschedprio(typedef unsigned long int pthread_t,
2188 int);

```

11.7.2 semaphore.h

```

2189
2190 extern int sem_close(sem_t *);
2191 extern int sem_destroy(sem_t *);
2192 extern int sem_getvalue(sem_t *, int *);
2193 extern int sem_init(sem_t *, int, unsigned int);
2194 extern sem_t *sem_open(const char *, int, ...);
2195 extern int sem_post(sem_t *);
2196 extern int sem_trywait(sem_t *);
2197 extern int sem_unlink(const char *);
2198 extern int sem_wait(sem_t *);
2199 extern int sem_timedwait(sem_t *, const struct timespec *);

```

11.8 Interfaces for libgcc_s

2200 Table 11-31 defines the library name and shared object name for the libgcc_s library

2201 **Table 11-31 libgcc_s Definition**

Library:	libgcc_s
SONAME:	libgcc_s.so.1

2203 The behavior of the interfaces in this library is specified by the following specifica-
2204 tions:

2205 [LSB] This Specification

11.8.1 Unwind Library

11.8.1.1 Interfaces for Unwind Library

2206 An LSB conforming implementation shall provide the architecture specific functions
2207 for Unwind Library specified in Table 11-32, with the full mandatory functionality as
2208 described in the referenced underlying specification.

2210 **Table 11-32 libgcc_s - Unwind Library Function Interfaces**

_Unwind_Backtra ce(GCC_3.3) [LSB]	_Unwind_DeleteE xception(GCC_3.0) [LSB]	_Unwind_FindEn closingFunction(G CC_3.3) [LSB]	_Unwind_Find_F DE(GCC_3.0) [LSB]
_Unwind_Forced Unwind(GCC_3.0) [LSB]	_Unwind_GetCF A(GCC_3.3) [LSB]	_Unwind_GetDat aRelBase(GCC_3. 0) [LSB]	_Unwind_GetGR(GCC_3.0) [LSB]
_Unwind_GetIP(GCC_3.0) [LSB]	_Unwind_GetLan guageSpecificDat a(GCC_3.0) [LSB]	_Unwind_GetReg ionStart(GCC_3.0) [LSB]	_Unwind_GetText RelBase(GCC_3.0) [LSB]
_Unwind_RaiseEx ception(GCC_3.0) [LSB]	_Unwind_Resum e(GCC_3.0) [LSB]	_Unwind_Resum e_or_Rethrow(GC C_3.3) [LSB]	_Unwind_SetGR(GCC_3.0) [LSB]

_Unwind_SetIP(G CC_3.0) [LSB]			
----------------------------------	--	--	--

2211

11.9 Data Definitions for libgcc_s

2212 This section defines global identifiers and their values that are associated with
2213 interfaces contained in libgcc_s. These definitions are organized into groups that
2214 correspond to system headers. This convention is used as a convenience for the
2215 reader, and does not imply the existence of these headers, or their content. Where an
2216 interface is defined as requiring a particular system header file all of the data
2217 definitions for that system header file presented here shall be in effect.

2218 This section gives data definitions to promote binary application portability, not to
2219 repeat source interface definitions available elsewhere. System providers and
2220 application developers should use this ABI to supplement - not to replace - source
2221 interface definition specifications.

2222 This specification uses the ISO C (1999) C Language as the reference programming
2223 language, and data definitions are specified in ISO C format. The C language is used
2224 here as a convenient notation. Using a C language description of these data objects
2225 does not preclude their use by other programming languages.

11.9.1 unwind.h

```

2226
2227 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2228 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2229 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2230 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2231                                         _Unwind_Stop_Fn, void *);
2232 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2233 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2234 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2235 _Unwind_Context
2236                                         *);
2237 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2238 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2239 _Unwind_Exception
2240                                         *);
2241 extern void _Unwind_Resume(struct _Unwind_Exception *);
2242 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2243 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2244 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2245 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2246 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2247                                         _Unwind_Stop_Fn, void *);
2248 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2249 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2250 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2251 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2252 _Unwind_Context
2253                                         *);
2254 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2255 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2256 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2257 _Unwind_Exception
2258                                         *);
2259 extern void _Unwind_Resume(struct _Unwind_Exception *);
2260 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2261 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2262 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,

```

```

2263                                     _Unwind_Stop_Fn, void *);
2264 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2265 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2266 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2267 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2268 _Unwind_Context
2269                                     *);
2270 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2271 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2272 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2273 _Unwind_Exception
2274                                     *);
2275 extern void _Unwind_Resume(struct _Unwind_Exception *);
2276 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2277 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2278 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2279 extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2280 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2281 _Unwind_Stop_Fn, void *);
2282 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2283 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2284 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2285 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2286 _Unwind_Context
2287                                     *);
2288 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2289 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2290 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2291 _Unwind_Exception
2292                                     *);
2293 extern void _Unwind_Resume(struct _Unwind_Exception *);
2294 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2295 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2296 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2297 extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2298 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2299 _Unwind_Stop_Fn, void *);
2300 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2301 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2302 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2303 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2304 _Unwind_Context
2305                                     *);
2306 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2307 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2308 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2309 _Unwind_Exception
2310                                     *);
2311 extern void _Unwind_Resume(struct _Unwind_Exception *);
2312 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2313 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2314 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2315 extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2316 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2317 _Unwind_Stop_Fn, void *);
2318 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2319 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2320 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2321 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2322 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2323 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2324 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2325 _Unwind_Exception
2326                                     *);

```

```

2327     extern void _Unwind_Resume(struct _Unwind_Exception *);
2328     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2329     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2330     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2331     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2332     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2333                                             _Unwind_Stop_Fn, void *);
2334     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2335     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2336     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2337     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2338     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2339     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2340     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2341     _Unwind_Exception
2342     *);
2343     extern void _Unwind_Resume(struct _Unwind_Exception *);
2344     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2345     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2346     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2347     *);
2348     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2349     *);
2350     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2351     *);
2352     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2353     *);
2354     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2355     *);
2356     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2357     *);
2358     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2359     *);
2360     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2361     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2362     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2363     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2364     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2365     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2366     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2367     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2368     _Unwind_Exception *);
2369     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2370     _Unwind_Exception *);
2371     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2372     _Unwind_Exception *);
2373     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2374     _Unwind_Exception *);
2375     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2376     _Unwind_Exception *);
2377     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2378     _Unwind_Exception *);
2379     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2380     _Unwind_Exception *);
2381     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2382     _Unwind_Exception *);
2383     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2384     _Unwind_Exception *);
2385     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2386     _Unwind_Exception *);
2387     extern void *_Unwind_FindEnclosingFunction(void *);
2388     extern void *_Unwind_FindEnclosingFunction(void *);
2389     extern void *_Unwind_FindEnclosingFunction(void *);

```

```

2391     extern void *_Unwind_FindEnclosingFunction(void *);
2392     extern void *_Unwind_FindEnclosingFunction(void *);
2393     extern void *_Unwind_FindEnclosingFunction(void *);
2394     extern void *_Unwind_FindEnclosingFunction(void *);
2395     extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);

```

11.10 Interface Definitions for libgcc_s

2396 The interfaces defined on the following pages are included in libgcc_s and are
 2397 defined by this specification. Unless otherwise noted, these interfaces shall be
 2398 included in the source standard.

2399 Other interfaces listed in Section 11.8 shall behave as described in the referenced
 2400 base document.

_Unwind_DeleteException

Name

2401 `_Unwind_DeleteException` – private C++ error handling method

Synopsis

```
2402 void _Unwind_DeleteException(struct _Unwind_Exception * object);
```

Description

2403 `_Unwind_DeleteException()` deletes the given exception *object*. If a given
 2404 runtime resumes normal execution after catching a foreign exception, it will not
 2405 know how to delete that exception. Such an exception shall be deleted by calling
 2406 `_Unwind_DeleteException()`. This is a convenience function that calls the function
 2407 pointed to by the *exception_cleanup* field of the exception header.

_Unwind_Find_FDE

Name

2408 `_Unwind_Find_FDE` – private C++ error handling method

Synopsis

```
2409 fde * _Unwind_Find_FDE(void * pc, struct dwarf_eh_bases * bases);
```

Description

2410 `_Unwind_Find_FDE()` looks for the object containing *pc*, then inserts into *bases*.

`_Unwind_ForcedUnwind`

Name

2411 `_Unwind_ForcedUnwind` — private C++ error handling method

Synopsis

```
2412 _Unwind_Reason_Code _Unwind_ForcedUnwind(struct _Unwind_Exception *  
2413 object, _Unwind_Stop_Fn stop, void * stop_parameter);
```

Description

2414 `_Unwind_ForcedUnwind()` raises an exception for forced unwinding, passing along
2415 the given exception *object*, which should have its *exception_class* and
2416 *exception_cleanup* fields set. The exception *object* has been allocated by the
2417 language-specific runtime, and has a language-specific format, except that it shall
2418 contain an `_Unwind_Exception` struct.

2419 Forced unwinding is a single-phase process. *stop* and *stop_parameter* control the
2420 termination of the unwind process instead of the usual personality routine query.
2421 *stop* is called for each unwind frame, with the parameters described for the usual
2422 personality routine below, plus an additional *stop_parameter*.

Return Value

2423 When *stop* identifies the destination frame, it transfers control to the user code as
2424 appropriate without returning, normally after calling `_Unwind_DeleteException()`.
2425 If not, then it should return an `_Unwind_Reason_Code` value.

2426 If *stop* returns any reason code other than `_URC_NO_REASON`, then the stack state is
2427 indeterminate from the point of view of the caller of `_Unwind_ForcedUnwind()`.
2428 Rather than attempt to return, therefore, the unwind library should use the
2429 *exception_cleanup* entry in the exception, and then call `abort()`.

2430 `_URC_NO_REASON`

2431 This is not the destination from. The unwind runtime will call frame's
2432 personality routine with the `_UA_FORCE_UNWIND` and `_UA_CLEANUP_PHASE` flag
2433 set in *actions*, and then unwind to the next frame and call the `stop()` function
2434 again.

2435 `_URC_END_OF_STACK`

2436 In order to allow `_Unwind_ForcedUnwind()` to perform special processing
2437 when it reaches the end of the stack, the unwind runtime will call it after the last
2438 frame is rejected, with a NULL stack pointer in the context, and the `stop()`
2439 function shall catch this condition. It may return this code if it cannot handle
2440 end-of-stack.

2441 `_URC_FATAL_PHASE2_ERROR`

2442 The `stop()` function may return this code for other fatal conditions like stack
2443 corruption.

_Unwind_GetDataRelBase

Name

2444 `_Unwind_GetDataRelBase` – private IA64 C++ error handling method

Synopsis

2445 `_Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context * context);`

Description

2446 `_Unwind_GetDataRelBase()` returns the global pointer in register one for *context*.

_Unwind_GetGR

Name

2447 `_Unwind_GetGR` – private C++ error handling method

Synopsis

2448 `_Unwind_Word _Unwind_GetGR(struct _Unwind_Context * context, int index);`

Description

2449 `_Unwind_GetGR()` returns data at *index* found in *context*. The register is identified
2450 by its index: 0 to 31 are for the fixed registers, and 32 to 127 are for the stacked
2451 registers.

2452 During the two phases of unwinding, only GR1 has a guaranteed value, which is the
2453 global pointer of the frame referenced by the unwind *context*. If the register has its
2454 NAT bit set, the behavior is unspecified.

_Unwind_GetIP

Name

2455 `_Unwind_GetIP` – private C++ error handling method

Synopsis

2456 `_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);`

Description

2457 `_Unwind_GetIP()` returns the instruction pointer value for the routine identified by
2458 the unwind *context*.

_Unwind_GetLanguageSpecificData**Name**

2459 `_Unwind_GetLanguageSpecificData` – private C++ error handling method

Synopsis

2460 `_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *
2461 context, uint value);`

Description

2462 `_Unwind_GetLanguageSpecificData()` returns the address of the language specific
2463 data area for the current stack frame.

_Unwind_GetRegionStart**Name**

2464 `_Unwind_GetRegionStart` – private C++ error handling method

Synopsis

2465 `_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);`

Description

2466 `_Unwind_GetRegionStart()` routine returns the address (i.e., 0) of the beginning of
2467 the procedure or code fragment described by the current unwind descriptor block.

_Unwind_GetTextRelBase**Name**

2468 `_Unwind_GetTextRelBase` – private IA64 C++ error handling method

Synopsis

2469 `_Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context * context);`

Description

2470 `_Unwind_GetTextRelBase()` calls the abort method, then returns.

_Unwind_RaiseException

Name

2471 `_Unwind_RaiseException` – private C++ error handling method

Synopsis

2472 `_Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *
2473 object);`

Description

2474 `_Unwind_RaiseException()` raises an exception, passing along the given exception
2475 *object*, which should have its *exception_class* and *exception_cleanup* fields set.
2476 The exception object has been allocated by the language-specific runtime, and has a
2477 language-specific format, exception that it shall contain an `_Unwind_Exception`.

Return Value

2478 `_Unwind_RaiseException()` does not return unless an error condition is found. If
2479 an error condition occurs, an `_Unwind_Reason_Code` is returned:

2480 `_URC_END_OF_STACK`

2481 The unwinder encountered the end of the stack during phase one without
2482 finding a handler. The unwind runtime will not have modified the stack. The
2483 C++ runtime will normally call `uncaught_exception()` in this case.

2484 `_URC_FATAL_PHASE1_ERROR`

2485 The unwinder encountered an unexpected error during phase one, because of
2486 something like stack corruption. The unwind runtime will not have modified
2487 the stack. The C++ runtime will normally call `terminate()` in this case.

2488 `_URC_FATAL_PHASE2_ERROR`

2489 The unwinder encountered an unexpected error during phase two. This is
2490 usually a *throw*, which will call `terminate()`.

_Unwind_Resume

Name

2491 `_Unwind_Resume` – private C++ error handling method

Synopsis

2492 `void _Unwind_Resume(struct _Unwind_Exception * object);`

Description

2493 `_Unwind_Resume()` resumes propagation of an existing exception *object*. A call to
2494 this routine is inserted as the end of a landing pad that performs cleanup, but does
2495 not resume normal execution. It causes unwinding to proceed further.

_Unwind_SetGR**Name**

2496 `_Unwind_SetGR` – private C++ error handling method

Synopsis

2497 `void _Unwind_SetGR(struct _Unwind_Context * context, int index, uint value);`

Description

2498 `_Unwind_SetGR()` sets the *value* of the register *indexed* for the routine identified by
2499 the unwind *context*.

_Unwind_SetIP**Name**

2500 `_Unwind_SetIP` – private C++ error handling method

Synopsis

2501 `void _Unwind_SetIP(struct _Unwind_Context * context, uint value);`

Description

2502 `_Unwind_SetIP()` sets the *value* of the instruction pointer for the routine identified
2503 by the unwind *context*

11.11 Interfaces for libdl

2504 Table 11-33 defines the library name and shared object name for the libdl library

2505 **Table 11-33 libdl Definition**

Library:	libdl
SONAME:	libdl.so.2

2506

2507 The behavior of the interfaces in this library is specified by the following specifica-
2508 tions:

[LSB] This Specification
[SUSv3] ISO POSIX (2003)

2509

11.11.1 Dynamic Loader**11.11.1.1 Interfaces for Dynamic Loader**

2510

2511 An LSB conforming implementation shall provide the architecture specific functions
2512 for Dynamic Loader specified in Table 11-34, with the full mandatory functionality
2513 as described in the referenced underlying specification.

2514 **Table 11-34 libdl - Dynamic Loader Function Interfaces**

<code>dladdr(GLIBC_2.2 .5) [LSB]</code>	<code>dlclose(GLIBC_2.2 .5) [SUSv3]</code>	<code>dLError(GLIBC_2. 2.5) [SUSv3]</code>	<code>dlopen(GLIBC_2. 2.5) [LSB]</code>
---	--	--	---

dlsym(GLIBC_2.2.5) [LSB]			
--------------------------	--	--	--

2515

11.12 Data Definitions for libdl

2516 This section defines global identifiers and their values that are associated with
 2517 interfaces contained in libdl. These definitions are organized into groups that
 2518 correspond to system headers. This convention is used as a convenience for the
 2519 reader, and does not imply the existence of these headers, or their content. Where an
 2520 interface is defined as requiring a particular system header file all of the data
 2521 definitions for that system header file presented here shall be in effect.

2522 This section gives data definitions to promote binary application portability, not to
 2523 repeat source interface definitions available elsewhere. System providers and
 2524 application developers should use this ABI to supplement - not to replace - source
 2525 interface definition specifications.

2526 This specification uses the ISO C (1999) C Language as the reference programming
 2527 language, and data definitions are specified in ISO C format. The C language is used
 2528 here as a convenient notation. Using a C language description of these data objects
 2529 does not preclude their use by other programming languages.

11.12.1 dlfcn.h

```
2530
2531 extern int dladdr(const void *, Dl_info *);
2532 extern int dlclose(void *);
2533 extern char *dlerror(void);
2534 extern void *dlopen(char *, int);
2535 extern void *dlsym(void *, char *);
```

11.13 Interfaces for libcrypt

2536 Table 11-35 defines the library name and shared object name for the libcrypt library

2537 **Table 11-35 libcrypt Definition**

Library:	libcrypt
SONAME:	libcrypt.so.1

2538

2539 The behavior of the interfaces in this library is specified by the following specifica-
 2540 tions:

2541 [SUSv3] ISO POSIX (2003)

11.13.1 Encryption

11.13.1.1 Interfaces for Encryption

2542 An LSB conforming implementation shall provide the architecture specific functions
 2543 for Encryption specified in Table 11-36, with the full mandatory functionality as
 2544 described in the referenced underlying specification.

2546 **Table 11-36 libcrypt - Encryption Function Interfaces**

crypt(GLIBC_2.2.5) [SUSv3]	encrypt(GLIBC_2.2.5) [SUSv3]	setkey(GLIBC_2.2.5) [SUSv3]	
----------------------------	------------------------------	-----------------------------	--

2547

IV Utility Libraries

12 Libraries

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

12.1 Interfaces for libz

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

6 **Table 12-1 libz Definition**

Library:	libz
SONAME:	libz.so.1

7

12.1.1 Compression Library

8 12.1.1.1 Interfaces for Compression Library

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

12.2 Data Definitions for libz

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

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

21 This specification uses the ISO C (1999) C Language as the reference programming
22 language, and data definitions are specified in ISO C . The C language is used here
23 as a convenient notation. Using a C language description of these data objects does
24 not preclude their use by other programming languages.

12.2.1 zlib.h

```
25  
26       extern int gzread(gzFile, voidp, unsigned int);  
27       extern int gzclose(gzFile);  
28       extern gzFile gzopen(const char *, const char *);  
29       extern gzFile gzdopen(int, const char *);  
30       extern int gzwrite(gzFile, voidpc, unsigned int);  
31       extern int gzflush(gzFile, int);  
32       extern const char *gzerror(gzFile, int *);  
33       extern uLong Adler32(uLong, const Bytef *, uInt);  
34       extern int compress(Bytef *, uLongf *, const Bytef *, uLong);  
35       extern int compress2(Bytef *, uLongf *, const Bytef *, uLong, int);  
36       extern uLong crc32(uLong, const Bytef *, uInt);  
37       extern int deflate(z_streamp, int);
```

```

38     extern int deflateCopy(z_streamp, z_streamp);
39     extern int deflateEnd(z_streamp);
40     extern int deflateInit2_(z_streamp, int, int, int, int, int, const char
41     *,
42         int);
43     extern int deflateInit_(z_streamp, int, const char *, int);
44     extern int deflateParams(z_streamp, int, int);
45     extern int deflateReset(z_streamp);
46     extern int deflateSetDictionary(z_streamp, const Bytef *, uInt);
47     extern const uLongf *get_crc_table(void);
48     extern int gzeof(gzFile);
49     extern int gzgetc(gzFile);
50     extern char *gzgets(gzFile, char *, int);
51     extern int gzprintf(gzFile, const char *, ...);
52     extern int gzputc(gzFile, int);
53     extern int gzputs(gzFile, const char *);
54     extern int gzrewind(gzFile);
55     extern z_off_t gzseek(gzFile, z_off_t, int);
56     extern int gzsetparams(gzFile, int, int);
57     extern z_off_t gztell(gzFile);
58     extern int inflate(z_streamp, int);
59     extern int inflateEnd(z_streamp);
60     extern int inflateInit2_(z_streamp, int, const char *, int);
61     extern int inflateInit_(z_streamp, const char *, int);
62     extern int inflateReset(z_streamp);
63     extern int inflateSetDictionary(z_streamp, const Bytef *, uInt);
64     extern int inflateSync(z_streamp);
65     extern int inflateSyncPoint(z_streamp);
66     extern int uncompress(Bytef *, uLongf *, const Bytef *, uLong);
67     extern const char *zError(int);
68     extern const char *zlibVersion(void);
69     extern uLong deflateBound(z_streamp, uLong);
70     extern uLong compressBound(uLong);

```

12.3 Interfaces for libncurses

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

72 **Table 12-2 libncurses Definition**

73 Library:	libncurses
SONAME:	libncurses.so.5

12.3.1 Curses

74 12.3.1.1 Interfaces for Curses

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

12.4 Data Definitions for libncurses

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

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

87 This specification uses the ISO C (1999) C Language as the reference programming
 88 language, and data definitions are specified in ISO C. The C language is used here
 89 as a convenient notation. Using a C language description of these data objects does
 90 not preclude their use by other programming languages.

12.4.1 curses.h

```

91
92 extern int addch(const chtype);
93 extern int addchnstr(const chtype *, int);
94 extern int addchstr(const chtype *);
95 extern int addnstr(const char *, int);
96 extern int addstr(const char *);
97 extern int attroff(int);
98 extern int attron(int);
99 extern int attrset(int);
100 extern int attr_get(attr_t *, short *, void *);
101 extern int attr_off(attr_t, void *);
102 extern int attr_on(attr_t, void *);
103 extern int attr_set(attr_t, short, void *);
104 extern int baudrate(void);
105 extern int beep(void);
106 extern int bkgd(chtype);
107 extern void bkgdset(chtype);
108 extern int border(chtype, chtype, chtype, chtype, chtype, chtype,
109 chtype,
110 chtype);
111 extern int box(WINDOW *, chtype, chtype);
112 extern bool can_change_color(void);
113 extern int cbreak(void);
114 extern int chgat(int, attr_t, short, const void *);
115 extern int clear(void);
116 extern int clearok(WINDOW *, bool);
117 extern int clrtoeol(void);
118 extern int clrtoeol(void);
119 extern int color_content(short, short *, short *, short *);
120 extern int color_set(short, void *);
121 extern int copywin(const WINDOW *, WINDOW *, int, int, int, int, int,
122 int,
123 int);
124 extern int curs_set(int);
125 extern int def_prog_mode(void);
126 extern int def_shell_mode(void);
127 extern int delay_output(int);
128 extern int delch(void);
129 extern void delscreen(SCREEN *);
130 extern int delwin(WINDOW *);
131 extern int deleteln(void);
132 extern WINDOW *derwin(WINDOW *, int, int, int, int);
133 extern int doupdate(void);
134 extern WINDOW *dupwin(WINDOW *);
135 extern int echo(void);
136 extern int echochar(const chtype);
137 extern int erase(void);
138 extern int endwin(void);
139 extern char erasechar(void);
140 extern void filter(void);
141 extern int flash(void);

```

```

142     extern int flushing(void);
143     extern chtype getbkgd(WINDOW *);
144     extern int getch(void);
145     extern int getnstr(char *, int);
146     extern int getstr(char *);
147     extern WINDOW *getwin(FILE *);
148     extern int halfdelay(int);
149     extern bool has_colors(void);
150     extern bool has_ic(void);
151     extern bool has_il(void);
152     extern int hline(chtype, int);
153     extern void idcok(WINDOW *, bool);
154     extern int idlok(WINDOW *, bool);
155     extern void immedok(WINDOW *, bool);
156     extern chtype inch(void);
157     extern int inchnstr(chtype *, int);
158     extern int inchstr(chtype *);
159     extern WINDOW *initscr(void);
160     extern int init_color(short, short, short, short);
161     extern int init_pair(short, short, short);
162     extern int innstr(char *, int);
163     extern int insch(chtype);
164     extern int insdelln(int);
165     extern int insertln(void);
166     extern int insnstr(const char *, int);
167     extern int insstr(const char *);
168     extern int instr(char *);
169     extern int intrflush(WINDOW *, bool);
170     extern bool isendwin(void);
171     extern bool is_linetouched(WINDOW *, int);
172     extern bool is_wintouched(WINDOW *);
173     extern const char *keyname(int);
174     extern int keypad(WINDOW *, bool);
175     extern char killchar(void);
176     extern int leaveok(WINDOW *, bool);
177     extern char *longname(void);
178     extern int meta(WINDOW *, bool);
179     extern int move(int, int);
180     extern int mvaddch(int, int, const chtype);
181     extern int mvaddchnstr(int, int, const chtype *, int);
182     extern int mvaddchstr(int, int, const chtype *);
183     extern int mvaddnstr(int, int, const char *, int);
184     extern int mvaddstr(int, int, const char *);
185     extern int mvchgat(int, int, int, attr_t, short, const void *);
186     extern int mvcur(int, int, int, int);
187     extern int mvdelch(int, int);
188     extern int mvderwin(WINDOW *, int, int);
189     extern int mvgetch(int, int);
190     extern int mvgetnstr(int, int, char *, int);
191     extern int mvgetstr(int, int, char *);
192     extern int mvhline(int, int, chtype, int);
193     extern chtype mvinch(int, int);
194     extern int mvinchnstr(int, int, chtype *, int);
195     extern int mvinchstr(int, int, chtype *);
196     extern int mvinnstr(int, int, char *, int);
197     extern int mvinsch(int, int, chtype);
198     extern int mvinsnstr(int, int, const char *, int);
199     extern int mvinsstr(int, int, const char *);
200     extern int mvinstr(int, int, char *);
201     extern int mvprintw(int, int, char *, ...);
202     extern int mvscanw(int, int, const char *, ...);
203     extern int mvvline(int, int, chtype, int);
204     extern int mvwaddch(WINDOW *, int, int, const chtype);
205     extern int mvwaddchnstr(WINDOW *, int, int, const chtype *, int);

```

```

206     extern int mvwaddchstr(WINDOW *, int, int, const chtype *);
207     extern int mvwaddnstr(WINDOW *, int, int, const char *, int);
208     extern int mvwaddstr(WINDOW *, int, int, const char *);
209     extern int mvwchgat(WINDOW *, int, int, int, attr_t, short, const void
210     *);
211     extern int mvwdelch(WINDOW *, int, int);
212     extern int mvwgetch(WINDOW *, int, int);
213     extern int mvwgetnstr(WINDOW *, int, int, char *, int);
214     extern int mvwgetstr(WINDOW *, int, int, char *);
215     extern int mvwhline(WINDOW *, int, int, chtype, int);
216     extern int mvwin(WINDOW *, int, int);
217     extern chtype mvwinch(WINDOW *, int, int);
218     extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
219     extern int mvwinchstr(WINDOW *, int, int, chtype *);
220     extern int mvwinnstr(WINDOW *, int, int, char *, int);
221     extern int mvwinsch(WINDOW *, int, int, chtype);
222     extern int mvwinsnstr(WINDOW *, int, int, const char *, int);
223     extern int mvwinsstr(WINDOW *, int, int, const char *);
224     extern int mvwinstr(WINDOW *, int, int, char *);
225     extern int mvwprintw(WINDOW *, int, int, char *, ...);
226     extern int mvwscanw(WINDOW *, int, int, const char *, ...);
227     extern int mvwvline(WINDOW *, int, int, chtype, int);
228     extern int napms(int);
229     extern WINDOW *newpad(int, int);
230     extern SCREEN *newterm(const char *, FILE *, FILE *);
231     extern WINDOW *newwin(int, int, int, int);
232     extern int nl(void);
233     extern int nocbreak(void);
234     extern int nodelay(WINDOW *, bool);
235     extern int noecho(void);
236     extern int nonl(void);
237     extern void noqiflush(void);
238     extern int noraw(void);
239     extern int notimeout(WINDOW *, bool);
240     extern int overlay(const WINDOW *, WINDOW *);
241     extern int overwrite(const WINDOW *, WINDOW *);
242     extern int pair_content(short, short *, short *);
243     extern int pechochar(WINDOW *, chtype);
244     extern int pnoutrefresh(WINDOW *, int, int, int, int, int, int);
245     extern int prefresh(WINDOW *, int, int, int, int, int, int);
246     extern int printw(char *, ...);
247     extern int putwin(WINDOW *, FILE *);
248     extern void qiflush(void);
249     extern int raw(void);
250     extern int redrawwin(WINDOW *);
251     extern int refresh(void);
252     extern int resetty(void);
253     extern int reset_prog_mode(void);
254     extern int reset_shell_mode(void);
255     extern int ripoffline(int, int (*init) (WINDOW *, int)
256     );
257     extern int savetty(void);
258     extern int scanw(const char *, ...);
259     extern int scr_dump(const char *);
260     extern int scr_init(const char *);
261     extern int scrl(int);
262     extern int scroll(WINDOW *);
263     extern int scrollok(WINDOW *, typedef unsigned char bool);
264     extern int scr_restore(const char *);
265     extern int scr_set(const char *);
266     extern int setscreg(int, int);
267     extern SCREEN *set_term(SCREEN *);
268     extern int slk_attroff(const typedef unsigned long int chtype);
269     extern int slk_attron(const typedef unsigned long int chtype);

```

```

270     extern int slk_attrset(const typedef unsigned long int chtype);
271     extern int slk_attr_set(const typedef chtype attr_t, short, void *);
272     extern int slk_clear(void);
273     extern int slk_color(short);
274     extern int slk_init(int);
275     extern char *slk_label(int);
276     extern int slk_noutrefresh(void);
277     extern int slk_refresh(void);
278     extern int slk_restore(void);
279     extern int slk_set(int, const char *, int);
280     extern int slk_touch(void);
281     extern int standout(void);
282     extern int standend(void);
283     extern int start_color(void);
284     extern WINDOW *subpad(WINDOW *, int, int, int, int);
285     extern WINDOW *subwin(WINDOW *, int, int, int, int);
286     extern int syncok(WINDOW *, typedef unsigned char bool);
287     extern typedef unsigned long int chtype termattrs(void);
288     extern char *termname(void);
289     extern void timeout(int);
290     extern int typeahead(int);
291     extern int ungetch(int);
292     extern int untouchwin(WINDOW *);
293     extern void use_env(typedef unsigned char bool);
294     extern int vidattr(typedef unsigned long int chtype);
295     extern int vidputs(typedef unsigned long int chtype,
296                       int (*vidputs_int) (int)
297                       );
298     extern int vline(typedef unsigned long int chtype, int);
299     extern int vwprintw(WINDOW *, char *, typedef void *va_list);
300     extern int vw_printw(WINDOW *, const char *, typedef void *va_list);
301     extern int wvscanw(WINDOW *, const char *, typedef void *va_list);
302     extern int vw_scanw(WINDOW *, const char *, typedef void *va_list);
303     extern int waddch(WINDOW *, const typedef unsigned long int chtype);
304     extern int waddchnstr(WINDOW *, const typedef unsigned long int chtype
305                          *,
306                          int);
307     extern int waddchstr(WINDOW *, const typedef unsigned long int chtype
308                          *);
309     extern int waddnstr(WINDOW *, const char *, int);
310     extern int waddstr(WINDOW *, const char *);
311     extern int wattron(WINDOW *, int);
312     extern int wattroff(WINDOW *, int);
313     extern int wattrset(WINDOW *, int);
314     extern int wattr_get(WINDOW *, attr_t *, short *, void *);
315     extern int wattr_on(WINDOW *, typedef chtype attr_t, void *);
316     extern int wattr_off(WINDOW *, typedef chtype attr_t, void *);
317     extern int wattr_set(WINDOW *, typedef chtype attr_t, short, void *);
318     extern int wbkgd(WINDOW *, typedef unsigned long int chtype);
319     extern void wbkgdset(WINDOW *, typedef unsigned long int chtype);
320     extern int wborder(WINDOW *, typedef unsigned long int chtype,
321                       typedef unsigned long int chtype,
322                       typedef unsigned long int chtype,
323                       typedef unsigned long int chtype,
324                       typedef unsigned long int chtype,
325                       typedef unsigned long int chtype,
326                       typedef unsigned long int chtype,
327                       typedef unsigned long int chtype);
328     extern int wchgat(WINDOW *, int, typedef chtype attr_t, short,
329                      const void *);
330     extern int wclear(WINDOW *);
331     extern int wclrtoebot(WINDOW *);
332     extern int wclrtoeol(WINDOW *);
333     extern int wcolor_set(WINDOW *, short, void *);

```

```

334     extern void wcursyncup(WINDOW *);
335     extern int wdelch(WINDOW *);
336     extern int wdeleteln(WINDOW *);
337     extern int wechochar(WINDOW *, const typedef unsigned long int chtype);
338     extern int werase(WINDOW *);
339     extern int wgetch(WINDOW *);
340     extern int wgetnstr(WINDOW *, char *, int);
341     extern int wgetstr(WINDOW *, char *);
342     extern int whline(WINDOW *, typedef unsigned long int chtype, int);
343     extern typedef unsigned long int chtype winch(WINDOW *);
344     extern int winchnstr(WINDOW *, chtype *, int);
345     extern int winchstr(WINDOW *, chtype *);
346     extern int winnstr(WINDOW *, char *, int);
347     extern int winsch(WINDOW *, typedef unsigned long int chtype);
348     extern int winsdelln(WINDOW *, int);
349     extern int winsertln(WINDOW *);
350     extern int winsnstr(WINDOW *, const char *, int);
351     extern int winsstr(WINDOW *, const char *);
352     extern int winstr(WINDOW *, char *);
353     extern int wmove(WINDOW *, int, int);
354     extern int wnoutrefresh(WINDOW *);
355     extern int wprintw(WINDOW *, char *, ...);
356     extern int wredrawln(WINDOW *, int, int);
357     extern int wrefresh(WINDOW *);
358     extern int wscanw(WINDOW *, const char *, ...);
359     extern int wscrl(WINDOW *, int);
360     extern int wsetscrreg(WINDOW *, int, int);
361     extern int wstandout(WINDOW *);
362     extern int wstandend(WINDOW *);
363     extern void wsyncdown(WINDOW *);
364     extern void wsyncup(WINDOW *);
365     extern void wtimeout(WINDOW *, int);
366     extern int wtouchln(WINDOW *, int, int, int);
367     extern int wvline(WINDOW *, typedef unsigned long int chtype, int);
368     extern char *unctrl(typedef unsigned long int chtype);
369     extern int COLORS(void);
370     extern int COLOR_PAIRS(void);
371     extern chtype acs_map(void);
372     extern WINDOW *curscr(void);
373     extern WINDOW *stdscr(void);
374     extern int COLS(void);
375     extern int LINES(void);
376     extern int touchline(WINDOW *, int, int);
377     extern int touchwin(WINDOW *);

```

12.4.2 term.h

```

378
379     extern int putp(const char *);
380     extern int tigetflag(const char *);
381     extern int tigetnum(const char *);
382     extern char *tigetstr(const char *);
383     extern char *tparm(const char *, ...);
384     extern TERMINAL *set_curterm(TERMINAL *);
385     extern int del_curterm(TERMINAL *);
386     extern int restartterm(char *, int, int *);
387     extern int setupterm(char *, int, int *);
388     extern char *tgetstr(char *, char **);
389     extern char *tgoto(const char *, int, int);
390     extern int tgetent(char *, const char *);
391     extern int tgetflag(char *);
392     extern int tgetnum(char *);
393     extern int tputs(const char *, int, int (*putcproc) (int)
394         );

```

395 extern TERMINAL *cur_term(void);

12.5 Interfaces for libutil

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

397 **Table 12-3 libutil Definition**

Library:	libutil
SONAME:	libutil.so.1

398

399 The behavior of the interfaces in this library is specified by the following specifica-
400 tions:

401 [LSB] This Specification

12.5.1 Utility Functions

12.5.1.1 Interfaces for Utility Functions

402 An LSB conforming implementation shall provide the architecture specific functions
403 for Utility Functions specified in Table 12-4, with the full mandatory functionality as
404 described in the referenced underlying specification.
405

406 **Table 12-4 libutil - Utility Functions Function Interfaces**

forkpty(GLIBC_2.2.5) [LSB]	login(GLIBC_2.2.5) [LSB]	login_tty(GLIBC_2.2.5) [LSB]	logout(GLIBC_2.2.5) [LSB]
logwtmp(GLIBC_2.2.5) [LSB]	openpty(GLIBC_2.2.5) [LSB]		

407

V Package Format and Installation

13 Software Installation

13.1 Package Dependencies

1 The LSB runtime environment shall provide the following dependencies.

2 `lsb-core-amd64`

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

5 These dependencies shall have a version of 3.0.

6 Other LSB modules may add additional dependencies; such dependencies shall
7 have the format `lsb-module-amd64`.

13.2 Package Architecture Considerations

8 All packages must specify an architecture of `x86_64`. An LSB runtime environment
9 must accept an architecture of `x86_64` even if the native architecture is different.

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

Annex A Alphabetical Listing of Interfaces

A.1 libgcc_s

1 The behavior of the interfaces in this library is specified by the following Standards.
2 This Specification [LSB]

3 **Table A-1 libgcc_s Function Interfaces**

_Unwind_Backtrace[LSB]	_Unwind_GetDataRelBase[LSB]	_Unwind_RaiseException[LSB]
_Unwind_DeleteException[LSB]	_Unwind_GetGR[LSB]	_Unwind_Resume[LSB]
_Unwind_FindEnclosingFunction[LSB]	_Unwind_GetIP[LSB]	_Unwind_Resume_or_Rethrow[LSB]
_Unwind_Find_FDE[LSB]	_Unwind_GetLanguageSpecificData[LSB]	_Unwind_SetGR[LSB]
_Unwind_ForcedUnwind[LSB]	_Unwind_GetRegionStart[LSB]	_Unwind_SetIP[LSB]
_Unwind_GetCFA[LSB]	_Unwind_GetTextRelBase[LSB]	

4

A.2 libm

5 The behavior of the interfaces in this library is specified by the following Standards.
6 ISO C (1999) [ISOC99]
7 ISO POSIX (2003) [SUSv3]

7 **Table A-2 libm Function Interfaces**

__fpclassify[ISOC99]	__signbit[ISOC99]	exp2[SUSv3]
----------------------	-------------------	-------------

8

Annex B GNU Free Documentation License (Informative)

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

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

B.1 PREAMBLE

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

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

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

B.2 APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

B.3 VERBATIM COPYING

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

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

B.4 COPYING IN QUANTITY

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

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

86 If you publish or distribute Opaque copies of the Document numbering more than
87 100, you must either include a machine-readable Transparent copy along with each

88 Opaque copy, or state in or with each Opaque copy a publicly-accessible
89 computer-network location containing a complete Transparent copy of the
90 Document, free of added material, which the general network-using public has
91 access to download anonymously at no charge using public-standard network
92 protocols. If you use the latter option, you must take reasonably prudent steps, when
93 you begin distribution of Opaque copies in quantity, to ensure that this Transparent
94 copy will remain thus accessible at the stated location until at least one year after the
95 last time you distribute an Opaque copy (directly or through your agents or
96 retailers) of that edition to the public.

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

B.5 MODIFICATIONS

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

- 106 A. Use in the Title Page (and on the covers, if any) a title distinct from that of the
107 Document, and from those of previous versions (which should, if there were
108 any, be listed in the History section of the Document). You may use the same
109 title as a previous version if the original publisher of that version gives
110 permission.
- 111 B. List on the Title Page, as authors, one or more persons or entities responsible
112 for authorship of the modifications in the Modified Version, together with at
113 least five of the principal authors of the Document (all of its principal authors,
114 if it has less than five).
- 115 C. State on the Title page the name of the publisher of the Modified Version, as
116 the publisher.
- 117 D. Preserve all the copyright notices of the Document.
- 118 E. Add an appropriate copyright notice for your modifications adjacent to the
119 other copyright notices.
- 120 F. Include, immediately after the copyright notices, a license notice giving the
121 public permission to use the Modified Version under the terms of this License,
122 in the form shown in the Addendum below.
- 123 G. Preserve in that license notice the full lists of Invariant Sections and required
124 Cover Texts given in the Document's license notice.
- 125 H. Include an unaltered copy of this License.
- 126 I. Preserve the section entitled "History", and its title, and add to it an item
127 stating at least the title, year, new authors, and publisher of the Modified
128 Version as given on the Title Page. If there is no section entitled "History" in
129 the Document, create one stating the title, year, authors, and publisher of the
130 Document as given on its Title Page, then add an item describing the Modified
131 Version as stated in the previous sentence.
- 132 J. Preserve the network location, if any, given in the Document for public access
133 to a Transparent copy of the Document, and likewise the network locations

- 134 given in the Document for previous versions it was based on. These may be
135 placed in the "History" section. You may omit a network location for a work
136 that was published at least four years before the Document itself, or if the
137 original publisher of the version it refers to gives permission.
- 138 K. In any section entitled "Acknowledgements" or "Dedications", preserve the
139 section's title, and preserve in the section all the substance and tone of each of
140 the contributor acknowledgements and/or dedications given therein.
- 141 L. Preserve all the Invariant Sections of the Document, unaltered in their text and
142 in their titles. Section numbers or the equivalent are not considered part of the
143 section titles.
- 144 M. Delete any section entitled "Endorsements". Such a section may not be
145 included in the Modified Version.
- 146 N. Do not retitle any existing section as "Endorsements" or to conflict in title with
147 any Invariant Section.

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

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

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

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

B.6 COMBINING DOCUMENTS

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

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

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

B.7 COLLECTIONS OF DOCUMENTS

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

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

B.8 AGGREGATION WITH INDEPENDENT WORKS

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

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

B.9 TRANSLATION

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

B.10 TERMINATION

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

B.11 FUTURE REVISIONS OF THIS LICENSE

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

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

B.12 How to use this License for your documents

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

233 Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or
234 modify this document under the terms of the GNU Free Documentation License, Version
235 1.1 or any later version published by the Free Software Foundation; with the Invariant
236 Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the
237 Back-Cover Texts being LIST. A copy of the license is included in the section entitled
238 "GNU Free Documentation License".

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

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