

Linux Standard Base Core Specification

for S390 3.01

Linux Standard Base Core Specification for S390 3.01

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 Normative References.....	10
3 Requirements 2.1 Normative References.....	10
3.1 Relevant Libraries 2.2 Informative References/Bibliography	13
3.2 LSB Implementation Conformance 3 Requirements.....	16
3.3 LSB Application Conformance 3.1 Relevant Libraries.....	16
4 Definitions 3.2 LSB Implementation Conformance.....	16
5 Terminology 3.3 LSB Application Conformance	17
6 Documentation Conventions 4 Definitions.....	19
II Executable and Linking Format (ELF) 5 Terminology.....	20
7 Introduction 6 Documentation Conventions	22
8 Low Level System Information II Executable and Linking Format (ELF).....	23
8.1 Machine Interface 7 Introduction	24
8.2 Function Calling Sequence 8 Low Level System Information.....	25
8.3 Operating System 1 Machine Interface	25
8.4 Process Initialization 8.2 Function Calling Sequence	26
8.5 Coding Examples 8.3 Operating System Interface	26
8.6 Debug Information 8.4 Process Initialization.....	27
9 Object Format 8.5 Coding Examples	27
9.1 Introduction 8.6 Debug Information	27
9.2 ELF Header 9 Object Format.....	28
9.3 Sections 9.1 Introduction.....	28
9.4 Symbol Table 9.2 ELF Header	28
9.5 Relocation 9.3 Sections.....	28
10 Program Loading and Dynamic Linking 9.4 Symbol Table.....	29
10.1 Introduction 9.5 Relocation.....	29
10.2 Program Loading and Dynamic Linking	30
10.3 Dynamic Linking 10.1 Introduction.....	30
III Base Libraries 10.2 Program Loading.....	30
11 Libraries 10.3 Dynamic Linking.....	30
11.1 Program Interpreter/Dynamic Linker III Base Libraries	31
11.2 Interfaces for libe 11 Libraries	32
11.3 Data Definitions for libe 11.1 Program Interpreter/Dynamic Linker.....	32
11.4 Interfaces for libm libc	32
11.5 Data Definitions for libm libc	59
11.6 Interfaces for libpthread libm	89
11.7 Interfaces for libgcc_s 11.5 Data Definitions for libm.....	97
11.8 Interface Definitions for libgcc_s 11.6 Interfaces for libpthread	103
11.9 Interfaces for libdl 11.7 Data Definitions for libpthread.....	108
11.10 Interfaces for libcrypt libgcc_s.....	112
IV Utility Libraries 11.9 Data Definitions for libgcc_s.....	113
12 Libraries 11.10 Interface Definitions for libgcc_s.....	117
12.1 Interfaces for libz libdl.....	122
12.2 Interfaces for libncurses 11.12 Data Definitions for libdl.....	123

12.11.13 Interfaces for libutil libcrypt	123
V Package Format and Installation	125
13 Software Installation 12 Libraries	126
13.1 Package Dependencies 12.1 Interfaces for libz.....	126
13.2 Package Architecture Considerations 12.2 Data Definitions for libz.....	126
A Alphabetical Listing of Interfaces 12.3 Interfaces for libncurses.....	127
A.1 libgcc_s 12.4 Data Definitions for libncurses.....	127
B GNU Free Documentation License 12.5 Interfaces for libutil.....	133
B.1 PREAMBLE	134
B.2 APPLICABILITY AND DEFINITIONS 13 Software Installation.....	135
B.3 VERBATIM COPYING 13.1 Package Dependencies	135
B.4 COPYING IN QUANTITY 13.2 Package Architecture Considerations.....	135
B.5 MODIFICATIONS	136
B.6 COMBINING DOCUMENTS A.1 libgcc_s	136
B.7 COLLECTIONS OF DOCUMENTS	137
(Informative)	137
B.8 AGGREGATION WITH INDEPENDENT WORKS B.1 PREAMBLE	137
B.9 TRANSLATION B.2 APPLICABILITY AND DEFINITIONS	137
B.10 TERMINATION B.3 VERBATIM COPYING	138
B.11 FUTURE REVISIONS OF THIS LICENSE B.4 COPYING IN QUANTITY.....	138
B.12 How to use this License for your documents B.5 MODIFICATIONS.....	139
B.6 COMBINING DOCUMENTS.....	140
B.7 COLLECTIONS OF DOCUMENTS.....	141
B.8 AGGREGATION WITH INDEPENDENT WORKS.....	141
B.9 TRANSLATION	141
B.10 TERMINATION	141
B.11 FUTURE REVISIONS OF THIS LICENSE	142
B.12 How to use this License for your documents.....	142

List of Tables

2-1 Normative References	10
3-1 Standard Library Names 2-2 Other References	13
9-1 ELF Special Sections 3-1 Standard Library Names	16
11-1 libc Definition 9-1 ELF Special Sections	28
11-2 libc—RPC Function Interfaces 9-2 Additional Special Sections	28
11-31 libc—System Calls Function Interfaces Definition	32
11-42 libc - Standard I/O RPC Function Interfaces	32
11-53 libc - Standard I/O Data System Calls Function Interfaces	34
11-64 libc - Signal Handling Standard I/O Function Interfaces	38
11-75 libc - Signal Handling Standard I/O Data Interfaces	40
11-86 libc - Localization Functions Signal Handling Function Interfaces	40
11-97 libc - Localization Functions Signal Handling Data Interfaces	41
11-108 libc - Socket Interface Localization Functions Function Interfaces	42
11-119 libc - Wide Characters Function Localization Functions Data Interfaces	42
11-1210 libc - String Functions Socket Interface Function Interfaces	43
11-1311 libc - IPC Functions Wide Characters Function Interfaces	44
11-1412 libc - Regular Expressions String Functions Function Interfaces	46
11-1513 libc - Character Type IPC Functions Function Interfaces	48
11-1614 libc - Time Manipulation Regular Expressions Function Interfaces	49
11-1715 libc - Time Manipulation Data Character Type Functions Function Interfaces	49
11-1816 libc - Terminal Interface Functions Time Manipulation Function Interfaces	50
11-1917 libc - System Database Interface Function Time Manipulation Data Interfaces	51
11-2018 libc - Language Support Terminal Interface Functions Function Interfaces	51
11-2119 libc - Large File Support System Database Interface Function Interfaces	52
11-2220 libc - Standard Library Language Support Function Interfaces	53
11-2321 libc - Standard Library Data Large File Support Function Interfaces	54
11-24 libm Definition 11-22 libc - Standard Library Function Interfaces	54
11-25 libm—Math Function 23 libc - Standard Library Data Interfaces	58
11-2624 libm—Math Data Interfaces Definition	90
11-27 libpthread Definition 11-25 libm - Math Function Interfaces	90
11-28 libpthread—Realtime Threads Function 26 libm - Math Data Interfaces	97
11-2927 libpthread—Posix Threads Function Interfaces Definition	103
11-3028 libpthread - Thread aware versions of libc interfaces Realtime Threads	104
11-31 libgcc_s Definition 11-29 libpthread - Posix Threads Function Interfaces	104
11-32 libgcc_s—Unwind Library 30 libpthread - Thread aware versions of libc	107
11-33 libdl 31 libgcc_s Definition	112
11-34 libdl—Dynamic Loader 32 libgcc_s - Unwind Library Function Interfaces	113
11-35 libcrypt 33 libdl Definition	122
11-36 libcrypt—Encryption 34 libdl - Dynamic Loader Function Interfaces	122
12-1 libz 11-35 libcrypt Definition	123
12-2 libncurses Definition 11-36 libcrypt - Encryption Function Interfaces	124
12-3 libutil 11 libz Definition	126
12-4 libutil—Utility Functions Function Interfaces 12-2 libncurses Definition	127
A-1 libgcc_s Function Interfaces 12-3 libutil Definition	133
12-4 libutil - Utility Functions Function Interfaces	133
A-1 libgcc_s Function Interfaces	136

Foreword

1 | This is version 3.01 of the Linux Standard Base Core Specification for S390. 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 specification-supplement ("LSB-arch" or "archLSB") describing the parts of the
9 interface that vary by processor architecture. Together, the LSB-generic and the
10 architecture-specific supplement for a single hardware architecture provide a
11 complete interface specification for compiled application programs on systems that
12 share a common 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 S390 architecture specific Core module of the Linux Standards Base (LSB).
29 This module supplements the generic LSB Core module with those interfaces that
30 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 Normative References

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

2 References

2.1 Normative References

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

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

Table 2-1 Normative References

Name	Title	URL
DWARF Debugging Information Format, Revision 2.0.0	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf
DWARF Debugging Information Format, Revision 3.0.0 (Draft)	DWARF Debugging Information Format, Revision 3.0.0 (Draft)	http://refspecs.freestandards.org/dwarf/
Enterprise Systems Architecture/390 Principles of Operation	Enterprise Systems Architecture/390 Principles of Operation	http://oss.software.ibm.com/linux390/documentation-2.2.shtml
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 --C	
ISO POSIX (2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions	http://www.unix.org/version3/

Name	Title	URL
	ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004	
ISO/IEC TR14652	ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions	
ITU T-V.42	International Telecommunication Union Recommendation V.42 (2002): Error correcting procedures for DCEs using asynchronous to synchronous conversion ITU	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
Li18nux Globalization Specification	LI18NUNIX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUNIX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt
LINUX for S/390 ELF Application Binary Interface Supplement	LINUX for S/390 ELF Application Binary Interface Supplement	http://oss.software.ibm.com/linux390/documentation-2.2.shtml

2 *Normative* References

Name	Title	URL
PAM	Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar & R.Schemers (SunSoft)	http://www.opengroup.org/tech/rfc/mirror/rfc/rfc86.0.txt
RFC 1321: The MD5 Message Digest Algorithm	IETF RFC 1321: The MD5 Message Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 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
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	

Name	Title	URL
	Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989.(ISBN 0201566524)	
SVID Issue 4	System V Interface Definition,Fourth Edition	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
this specification	Linux Standard Base	http://www.linuxbase.org/spec/
X/Open Courses	CAE Specification, May 1996, X/Open Courses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	http://www.opengroup.org/publications/catalog/un.htm

2.2 Informative References/Bibliography

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

Table 2-2 Other References

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

2 ~~Normative~~ References

Name	Title	URL
	asynchronous-to-synchronous conversion ITUV	
Li18nux Globalization Specification	LI18NUNIX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUNIX-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-form

Name	Title	URL
		at-rpm-file-format.html
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

3 Requirements

3.1 Relevant Libraries

1 The libraries listed in Table 3-1 shall be available on S390 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	/lib/ld-lsb-s390.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 • ~~The implementation shall implement fully the architecture described in the~~
17 ~~hardware manual for the target processor architecture.~~
- 18 • A processor architecture represents a family of related processors which may not
19 have identical feature sets. The architecture specific supplement to this
20 specification for a given target processor architecture describes a minimum
21 acceptable processor. The implementation shall provide all features of this
22 processor, whether in hardware or through emulation transparent to the
23 application.
- 24 • The implementation shall be capable of executing compiled applications having
25 the format and using the system interfaces described in this document.

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

3.3 LSB Application Conformance

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

56 A conforming application shall satisfy the following requirements:

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

3 Requirements

- 68 | • If such an interface or data format is supplied by another application through
69 | direct invocation of that application during execution, that application ~~is~~shall be
70 | in turn an LSB conforming application.
 - 71 | • The use of that interface or data format, as well as its source, ~~is~~shall be identified
72 | in the documentation of the application.
 - 73 | • It shall not use any values for a named interface that are reserved for vendor
74 | extensions.
- 75 | A strictly conforming application ~~does~~shall not require or use any interface, facility,
76 | or implementation-defined extension that is not defined in this document in order to
77 | be 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.

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) [1SUSv3]`

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

25 **Note:** Symbol versions are defined in the ~~listed references below the~~
26 ~~table~~-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 LINUX for S/390 ELF Application Binary Interface Supplement, and is
4 intended to document additions made since the publication of that document.

8 Low Level System Information

8.1 Machine Interface

8.1.1 Processor Architecture

The ESA/390 Architecture is specified by the following documents

- LINUX for S/390 ELF Application Binary Interface Supplement
- Enterprise Systems Architecture/390 Principles of Operation

Only the features of ESA/390 processor instruction set and the following optional instructions may be assumed to be present:

- additional floating point facility
- compare and move extended facility
- immediate and relative instruction facility
- string instruction facility
- square-root facility

An application should determine if any additional instruction set features are available before using those additional features. If a feature is not present, then ~~the a conforming~~ application ~~may shall~~ not use it.

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

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

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

Applications conforming to this specification should attempt to execute in a diminished capacity if a required instruction set feature is not present.

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

8.1.2 Data Representation

LSB-conforming applications shall use the data representation as defined in Chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.1.2.1 Byte Ordering

See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.1.2.2 Fundamental Types

See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

34 **8.1.2.3 Aggregates and Unions**
35 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

36 **8.1.2.4 Bit Fields**
37 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.2 Function Calling Sequence

38 LSB-conforming applications shall use the function calling sequence as defined in
39 Chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.2.1 Registers
40 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.2.2 Stack Frame
41 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.2.3 Parameter Passing
42 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.2.4 Variable Argument Lists
43 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.2.5 Return Values
44 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.3 Operating System Interface

45 LSB-conforming applications shall use the Operating System Interfaces as defined in
46 Chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.3.1 Virtual Address Space
47 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.3.1.1 Page Size
48 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.
49

8.3.1.2 Virtual Address Assignments
50 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.
51

8.3.1.3 Managing the Process Stack
52 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.
53

8.3.1.4 Coding Guidelines
54 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.
55

8.3.2 Processor Execution Mode
56 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.3.3 Exception Interface

57 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.4 Process Initialization

58 LSB-conforming applications shall use the Process Initialization as defined in
59 Chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.4.1 Registers

60 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.4.2 Process Stack

61 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.5 Coding Examples

62 LSB-conforming applications may implement fundamental operations using the
63 Coding Examples as defined in Chapter 1 of the LINUX for S/390 ELF Application
64 Binary Interface Supplement.

8.5.1 Code Model Overview

65 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.5.2 Function Prolog and Epilog

66 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.5.3 Data Objects

67 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.5.4 Function Calls

68 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.5.5 Branching

69 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.5.6 Dynamic Stack Space Allocation

70 See chapter 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

8.6 Debug Information

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

9 Object Format

9.1 Introduction

1 LSB-conforming implementations shall support an object file , called Executable and
2 Linking Format (ELF) as defined by the System V ABI , System V ABI Update,
3 LINUX for S/390 ELF Application Binary Interface Supplement and as
4 supplemented by the ~~this specification~~ **This Specification** and this document.

9.2 ELF Header

9.2.1 Machine Information

5 LSB-conforming applications shall use the Machine Information as defined in
6 Chapter 2 of the LINUX for S/390 ELF Application Binary Interface Supplement.

9.3 Sections

7 See chapter 2 of the LINUX for S/390 ELF Application Binary Interface Supplement.

9.3.1 Special Sections

8 The following sections are defined in the LINUX for S/390 ELF Application Binary
9 Interface Supplement.

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

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

11
12 .got
13 This section holds the global offset table

14 .plt
15 This section holds the Procedure Linkage Table

9.3.2 Additional Special Sections

16 The following additional sections are defined here.

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

Name	Type	Attributes
.rela.dyn	SHT_RELA	SHF_ALLOC
.rela.plt	SHT_RELA	SHF_ALLOC

18
19 .rela.dyn
20 This section holds RELA type relocation information for all sections of a shared
21 library except the PLT

22
23
24**.rela.plt**

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

9.4 Symbol Table

25
26

LSB-conforming applications shall use the Symbol Table as defined in Chapter 2 of the LINUX for S/390 ELF Application Binary Interface Supplement.

9.5 Relocation

27
28

LSB-conforming applications shall use Relocations as defined in Chapter 2 of the LINUX for S/390 ELF Application Binary Interface Supplement.

9.5.1 Relocation Types

29

See chapter 2 of the LINUX for S/390 ELF Application Binary Interface 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 LINUX for S/390
3 ELF Application Binary Interface Supplement and as supplemented by the the
4 generic LSB and this document. LSB-conforming implementations need not support
5 tags related functionality. LSB-conforming applications must not rely on tags related
6 functionality.

10.2 Program Loading

7 See chapter 3 of the LINUX for S/390 ELF Application Binary Interface Supplement.

10.3 Dynamic Linking

8 See chapter 3 of the LINUX for S/390 ELF Application Binary Interface Supplement.

10.3.1 Dynamic Section

9 The following dynamic entries are defined in the LINUX for S/390 ELF Application
10 Binary Interface Supplement.

11 DT_JMPREL

12 This entry is associated with a table of relocation entries for the procedure
13 linkage table. This entry is mandatory both for executable and shared object
14 files

15 DT_PLTGOT

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

10.3.2 Global Offset Table

18 See chapter 3 of the LINUX for S/390 ELF Application Binary Interface Supplement.

10.3.3 Shared Object Dependencies

19 See chapter 3 of the LINUX for S/390 ELF Application Binary Interface Supplement.

10.3.4 Function Addresses

20 See chapter 3 of the LINUX for S/390 ELF Application Binary Interface Supplement.

10.3.5 Procedure Linkage Table

21 See chapter 3 of the LINUX for S/390 ELF Application Binary Interface Supplement.

III Base Libraries

11 Libraries

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

4 Only those interfaces that are unique to the S390 platform are defined here. This
 5 section should be used in conjunction with the corresponding section in the Linux
 6 Standard Base Specification.

11.1 Program Interpreter/Dynamic Linker

7 The ~~LSB specifies the~~ Program Interpreter ~~shall~~ be /lib/ld-lsb-s390.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~~ This Specification
- [SUSv2] SUSv2
- [SUSv3] ISO POSIX (2003)
- [SVID.3] SVID Issue 3
- [SVID.4] SVID Issue 4

11.2.1 RPC

11.2.1.1 Interfaces for RPC

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

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

authnone_create(GLIBC_2.0) [1]	svc_getreqset(GLIBC_2.0) [2]	svcadp_create(GLIBC_2.0) [3]	xdr_int(GLIBC_2.0) [2]	xdr_u_long(GLIBC_2.0) [2]
clnt_create(GLIBC_2.0) [1]	svc_register(GLIBC_2.0) [3]	xdr_accepted_reply(GLIBC_2.0) [2]	xdr_long(GLIBC_2.0) [2]	xdr_u_short(GLIBC_2.0) [2]
clnt_percreateerror(GLIBC_2.0) [1]	svc_run(GLIBC_2.0) [3]	xdr_array(GLIBC_2.0) [2]	xdr_opaque(GLIBC_2.0) [2]	xdr_union(GLIBC_2.0) [2]
clnt_permon(G	svc_sendreply	xdr_bool(GLI	xdr_opaque_a	xdr_vector(G

LIBC_2.0)[1]	y(GLIBC_2.0)[3]	BC_2.0)[2]	uth(GLIBC_2.0)[2]	LIBC_2.0)[2]
clnt_perror(GLIBC_2.0)[1]	svcerr_auth(GLIBC_2.0)[2]	xdr_bytes(GLIBC_2.0)[2]	xdr_pointer(GLIBC_2.0)[2]	xdr_void(GLIBC_2.0)[2]
clnt_sprecreateerror(GLIBC_2.0)[1]	svcerr_decode(GLIBC_2.0)[2]	xdr_callhdr(GLIBC_2.0)[2]	xdr_reference(GLIBC_2.0)[2]	xdr_wrapstring(GLIBC_2.0)[2]
clnt_sperrno(GLIBC_2.0)[1]	svcerr_noproc(GLIBC_2.0)[2]	xdr_callmsg(GLIBC_2.0)[2]	xdr_rejected_reply(GLIBC_2.0)[2]	xdrmem_create(GLIBC_2.0)[2]
clnt_sperror(GLIBC_2.0)[1]	svcerr_noprog(GLIBC_2.0)[2]	xdr_char(GLIBC_2.0)[2]	xdr_replymsg(GLIBC_2.0)[2]	xdrrec_create(GLIBC_2.0)[2]
key_decryptsession(GLIBC_2.1)[2]	svcerr_progvers(GLIBC_2.0)[2]	xdr_double(GLIBC_2.0)[2]	xdr_short(GLIBC_2.0)[2]	xdrrec_eof(GLIBC_2.0)[2]
pmap_getport(GLIBC_2.0)[3]	svcerr_systemerr(GLIBC_2.0)[2]	xdr_enum(GLIBC_2.0)[2]	xdr_string(GLIBC_2.0)[2]	
pmap_set(GLIBC_2.0)[3]	svcerr_weakauth(GLIBC_2.0)[2]	xdr_float(GLIBC_2.0)[2]	xdr_u_char(GLIBC_2.0)[2]	
pmap_unset(GLIBC_2.0)[3]	svctcp_create(GLIBC_2.0)[3]	xdr_free(GLIBC_2.0)[2]	xdr_u_int(GLIBC_2.0)[3]	

19
20
21
22
23

Referenced Specification(s)

[1]- SVID Issue 4

[2]- SVID Issue 3

[3]- this specification

authnone_create(GLIBC_2.0)[SVID.4]	clnt_create(GLIBC_2.0)[SVID.4]	clnt_pcreateerror(GLIBC_2.0)[SVID.4]	clnt_permno(GLIBC_2.0)[SVID.4]
clnt_perror(GLIBC_2.0)[SVID.4]	clnt_sprecreateerror(GLIBC_2.0)[SVID.4]	clnt_sperrno(GLIBC_2.0)[SVID.4]	clnt_sperror(GLIBC_2.0)[SVID.4]
key_decryptsession(GLIBC_2.1)[SVID.3]	pmap_getport(GLIBC_2.0)[LSB]	pmap_set(GLIBC_2.0)[LSB]	pmap_unset(GLIBC_2.0)[LSB]
svc_getreqset(GLIBC_2.0)[SVID.3]	svc_register(GLIBC_2.0)[LSB]	svc_run(GLIBC_2.0)[LSB]	svc_sendreply(GLIBC_2.0)[LSB]
svcerr_auth(GLIBC_2.0)[SVID.3]	svcerr_decode(GLIBC_2.0)[SVID.3]	svcerr_noproc(GLIBC_2.0)[SVID.3]	svcerr_noprog(GLIBC_2.0)[SVID.3]
svcerr_progvers(GLIBC_2.0)[SVID.3]	svcerr_systemerr(GLIBC_2.0)[SVID.3]	svcerr_weakauth(GLIBC_2.0)[SVID.3]	svctcp_create(GLIBC_2.0)[SVID.3]

GLIBC_2.0) [SVID.3]	GLIBC_2.0) [SVID.3]	GLIBC_2.0) [SVID.3]	BC_2.0) [LSB]
svcdp_create(GLIBC_2.0) [LSB]	xdr_accepted_repl y(GLIBC_2.0) [SVID.3]	xdr_array(GLIBC_2.0) [SVID.3]	xdr_bool(GLIBC_2.0) [SVID.3]
xdr_bytes(GLIBC_2.0) [SVID.3]	xdr_callhdr(GLIBC_2.0) [SVID.3]	xdr_callmsg(GLIBC_2.0) [SVID.3]	xdr_char(GLIBC_2.0) [SVID.3]
xdr_double(GLIBC_2.0) [SVID.3]	xdr_enum(GLIBC_2.0) [SVID.3]	xdr_float(GLIBC_2.0) [SVID.3]	xdr_free(GLIBC_2.0) [SVID.3]
xdr_int(GLIBC_2.0) [SVID.3]	xdr_long(GLIBC_2.0) [SVID.3]	xdr_opaque(GLIBC_2.0) [SVID.3]	xdr_opaque_auth(GLIBC_2.0) [SVID.3]
xdr_pointer(GLIBC_2.0) [SVID.3]	xdr_reference(GLIBC_2.0) [SVID.3]	xdr_rejected_repl y(GLIBC_2.0) [SVID.3]	xdr_replymsg(GLIBC_2.0) [SVID.3]
xdr_short(GLIBC_2.0) [SVID.3]	xdr_string(GLIBC_2.0) [SVID.3]	xdr_u_char(GLIBC_2.0) [SVID.3]	xdr_u_int(GLIBC_2.0) [LSB]
xdr_u_long(GLIBC_2.0) [SVID.3]	xdr_u_short(GLIBC_2.0) [SVID.3]	xdr_union(GLIBC_2.0) [SVID.3]	xdr_vector(GLIBC_2.0) [SVID.3]
xdr_void(GLIBC_2.0) [SVID.3]	xdr_wrapstring(GLIBC_2.0) [SVID.3]	xdrmem_create(GLIBC_2.0) [SVID.3]	xdrrec_create(GLIBC_2.0) [SVID.3]
xdrrec_eof(GLIBC_2.0) [SVID.3]			

24

11.2.2 System Calls

25

11.2.2.1 Interfaces for System Calls

26

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.

27

28

29

Table 11-3 libc - System Calls Function Interfaces

__fxstat(GLIBC_2.0) [1]	fehmod(GLIBC_2.0) [2]	getwd(GLIBC_2.0) [2]	read(GLIBC_2.0) [2]	setrlimit(GLIBC_2.2) [2]
__getpgid(GLIBC_2.0) [1]	fehown(GLIBC_2.0) [2]	initgroups(GLIBC_2.0) [1]	readdir(GLIBC_2.0) [2]	setrlimit64(GLIBC_2.1) [3]
__lxstat(GLIBC_2.0) [1]	fentl(GLIBC_2.0) [1]	ioctl(GLIBC_2.0) [1]	readdir_r(GLIBC_2.0) [2]	setsid(GLIBC_2.0) [2]
__xmknod(GLIBC_2.0) [1]	fdatasync(GLIBC_2.0) [2]	kill(GLIBC_2.0) [1]	readlink(GLIBC_2.0) [2]	setuid(GLIBC_2.0) [2]
__xstat(GLIBC_2.0) [1]	flock(GLIBC_2.0) [1]	killpg(GLIBC_2.0) [2]	readv(GLIBC_2.0) [2]	sleep(GLIBC_2.0) [2]
access(GLIBC_2.0) [1]	fork(GLIBC_2.0) [1]	lchown(GLIBC_2.0) [1]	rename(GLIBC_2.0) [1]	statvfs(GLIBC_2.0) [1]

<code>_2.0)</code> [2]	<code>.0)</code> [2]	<code>C_2.0)</code> [2]	<code>C_2.0)</code> [2]	<code>_2.1)</code> [2]
<code>acct(GLIBC_2.0)</code> [1]	<code>fstatvfs(GLIBC_2.1)</code> [2]	<code>link(GLIBC_2.0)</code> [1]	<code>rmdir(GLIBC_2.0)</code> [2]	<code>stime(GLIBC_2.0)</code> [1]
<code>alarm(GLIBC_2.0)</code> [2]	<code>fsync(GLIBC_2.0)</code> [2]	<code>lockf(GLIBC_2.0)</code> [2]	<code>sbrk(GLIBC_2.0)</code> [4]	<code>symlink(GLIBC_2.0)</code> [2]
<code>brk(GLIBC_2.0)</code> [4]	<code>ftime(GLIBC_2.0)</code> [2]	<code>lseek(GLIBC_2.0)</code> [2]	<code>sched_get_priority_max(GLIBC_2.0)</code> [2]	<code>sync(GLIBC_2.0)</code> [2]
<code>chdir(GLIBC_2.0)</code> [2]	<code>ftruncate(GLIBC_2.0)</code> [2]	<code>mkdir(GLIBC_2.0)</code> [2]	<code>sched_get_priority_min(GLIBC_2.0)</code> [2]	<code>sysconf(GLIBC_2.0)</code> [2]
<code>chmod(GLIBC_2.0)</code> [2]	<code>getcontext(GLIBC_2.1)</code> [2]	<code>mknfio(GLIBC_2.0)</code> [2]	<code>sched_getparam(GLIBC_2.0)</code> [2]	<code>time(GLIBC_2.0)</code> [2]
<code>chown(GLIBC_2.1)</code> [2]	<code>getegid(GLIBC_2.0)</code> [2]	<code>mlock(GLIBC_2.0)</code> [2]	<code>sched_getscheduler(GLIBC_2.0)</code> [2]	<code>times(GLIBC_2.0)</code> [2]
<code>chroot(GLIBC_2.0)</code> [4]	<code>geteuid(GLIBC_2.0)</code> [2]	<code>mlockall(GLIBC_2.0)</code> [2]	<code>sched_rr_get_interval(GLIBC_2.0)</code> [2]	<code>truncate(GLIBC_2.0)</code> [2]
<code>clock(GLIBC_2.0)</code> [2]	<code>getgid(GLIBC_2.0)</code> [2]	<code>mmap(GLIBC_2.0)</code> [2]	<code>sched_setparam(GLIBC_2.0)</code> [2]	<code>ulimit(GLIBC_2.0)</code> [2]
<code>close(GLIBC_2.0)</code> [2]	<code>getgroups(GLIBC_2.0)</code> [2]	<code>mprotect(GLIBC_2.0)</code> [2]	<code>sched_setscheduler(GLIBC_2.0)</code> [2]	<code>umask(GLIBC_2.0)</code> [2]
<code>closedir(GLIBC_2.0)</code> [2]	<code>getitimer(GLIBC_2.0)</code> [2]	<code>msync(GLIBC_2.0)</code> [2]	<code>sched_yield(GLIBC_2.0)</code> [2]	<code>uname(GLIBC_2.0)</code> [2]
<code>creat(GLIBC_2.0)</code> [2]	<code>getloadavg(GLIBC_2.2)</code> [1]	<code>munlock(GLIBC_2.0)</code> [2]	<code>select(GLIBC_2.0)</code> [2]	<code>unlink(GLIBC_2.0)</code> [1]
<code>dup(GLIBC_2.0)</code> [2]	<code>getpagesize(GLIBC_2.0)</code> [4]	<code>munlockall(GLIBC_2.0)</code> [2]	<code>setcontext(GLIBC_2.0)</code> [2]	<code>utime(GLIBC_2.0)</code> [2]
<code>dup2(GLIBC_2.0)</code> [2]	<code>getpgid(GLIBC_2.0)</code> [2]	<code>munmap(GLIBC_2.0)</code> [2]	<code>setegid(GLIBC_2.0)</code> [2]	<code>utimes(GLIBC_2.0)</code> [2]
<code>execl(GLIBC_2.0)</code> [2]	<code>getpgrp(GLIBC_2.0)</code> [2]	<code>nanosleep(GLIBC_2.0)</code> [2]	<code>seteuid(GLIBC_2.0)</code> [2]	<code>vfork(GLIBC_2.0)</code> [2]
<code>execl(GLIBC_2.0)</code> [2]	<code>getpid(GLIBC_2.0)</code> [2]	<code>nice(GLIBC_2.0)</code> [2]	<code>setgid(GLIBC_2.0)</code> [2]	<code>wait(GLIBC_2.0)</code> [2]
<code>execlp(GLIBC_2.0)</code> [2]	<code>getppid(GLIBC_2.0)</code> [2]	<code>open(GLIBC_2.0)</code> [2]	<code>setitimer(GLIBC_2.0)</code> [2]	<code>wait4(GLIBC_2.0)</code> [1]
<code>execv(GLIBC_2.0)</code> [2]	<code>getpriority(GLIBC_2.0)</code> [2]	<code>opendir(GLIBC_2.0)</code> [2]	<code>setpgid(GLIBC_2.0)</code> [2]	<code>waitpid(GLIBC_2.0)</code> [1]

<code>execve</code> (GLIBC_2.0) [2]	<code>getrlimit</code> (GLIBC_2.2) [2]	<code>pathconf</code> (GLIBC_2.0) [2]	<code>setpgrp</code> (GLIBC_2.0) [2]	<code>write</code> (GLIBC_2.0) [2]
<code>execvp</code> (GLIBC_2.0) [2]	<code>getrusage</code> (GLIBC_2.0) [2]	<code>pause</code> (GLIBC_2.0) [2]	<code>setpriority</code> (GLIBC_2.0) [2]	<code>writew</code> (GLIBC_2.0) [2]
<code>exit</code> (GLIBC_2.0) [2]	<code>getsid</code> (GLIBC_2.0) [2]	<code>pipe</code> (GLIBC_2.0) [2]	<code>setregid</code> (GLIBC_2.0) [2]	
<code>fchdir</code> (GLIBC_2.0) [2]	<code>getuid</code> (GLIBC_2.0) [2]	<code>poll</code> (GLIBC_2.0) [2]	<code>setreuid</code> (GLIBC_2.0) [2]	

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

[3]. Large File Support

[4]. SUSv2

<code>__fxstat</code> (GLIBC_2.0) [LSB]	<code>__getpgid</code> (GLIBC_2.0) [LSB]	<code>__lxstat</code> (GLIBC_2.0) [LSB]	<code>__xmknod</code> (GLIBC_2.0) [LSB]
<code>__xstat</code> (GLIBC_2.0) [LSB]	<code>access</code> (GLIBC_2.0) [SUSv3]	<code>acct</code> (GLIBC_2.0) [LSB]	<code>alarm</code> (GLIBC_2.0) [SUSv3]
<code>brk</code> (GLIBC_2.0) [SUSv2]	<code>chdir</code> (GLIBC_2.0) [SUSv3]	<code>chmod</code> (GLIBC_2.0) [SUSv3]	<code>chown</code> (GLIBC_2.1) [SUSv3]
<code>chroot</code> (GLIBC_2.0) [SUSv2]	<code>clock</code> (GLIBC_2.0) [SUSv3]	<code>close</code> (GLIBC_2.0) [SUSv3]	<code>closedir</code> (GLIBC_2.0) [SUSv3]
<code>creat</code> (GLIBC_2.0) [SUSv3]	<code>dup</code> (GLIBC_2.0) [SUSv3]	<code>dup2</code> (GLIBC_2.0) [SUSv3]	<code>execl</code> (GLIBC_2.0) [SUSv3]
<code>execle</code> (GLIBC_2.0) [SUSv3]	<code>execlp</code> (GLIBC_2.0) [SUSv3]	<code>execv</code> (GLIBC_2.0) [SUSv3]	<code>execve</code> (GLIBC_2.0) [SUSv3]
<code>execvp</code> (GLIBC_2.0) [SUSv3]	<code>exit</code> (GLIBC_2.0) [SUSv3]	<code>fchdir</code> (GLIBC_2.0) [SUSv3]	<code>fchmod</code> (GLIBC_2.0) [SUSv3]
<code>fchown</code> (GLIBC_2.0) [SUSv3]	<code>fcntl</code> (GLIBC_2.0) [LSB]	<code>fdatasync</code> (GLIBC_2.0) [SUSv3]	<code>flock</code> (GLIBC_2.0) [LSB]
<code>fork</code> (GLIBC_2.0) [SUSv3]	<code>fstatvfs</code> (GLIBC_2.1) [SUSv3]	<code>fsync</code> (GLIBC_2.0) [SUSv3]	<code>ftime</code> (GLIBC_2.0) [SUSv3]
<code>ftruncate</code> (GLIBC_2.0) [SUSv3]	<code>getcontext</code> (GLIBC_2.1) [SUSv3]	<code>getegid</code> (GLIBC_2.0) [SUSv3]	<code>geteuid</code> (GLIBC_2.0) [SUSv3]
<code>getgid</code> (GLIBC_2.0) [SUSv3]	<code>getgroups</code> (GLIBC_2.0) [SUSv3]	<code>getitimer</code> (GLIBC_2.0) [SUSv3]	<code>getloadavg</code> (GLIBC_2.2) [LSB]
<code>getpagesize</code> (GLIBC_2.0) [SUSv2]	<code>getpgid</code> (GLIBC_2.0) [SUSv3]	<code>getpgrp</code> (GLIBC_2.0) [SUSv3]	<code>getpid</code> (GLIBC_2.0) [SUSv3]
<code>getppid</code> (GLIBC_2.0) [SUSv3]	<code>getpriority</code> (GLIBC_2.0) [SUSv3]	<code>getrlimit</code> (GLIBC_2.2) [SUSv3]	<code>getrusage</code> (GLIBC_2.0) [SUSv3]
<code>getsid</code> (GLIBC_2.0)	<code>getuid</code> (GLIBC_2.0)	<code>getwd</code> (GLIBC_2.0)	<code>initgroups</code> (GLIBC_2.0)

[SUSv3]) [SUSv3]) [SUSv3]	_2.0) [LSB]
ioctl(GLIBC_2.0) [LSB]	kill(GLIBC_2.0) [LSB]	killpg(GLIBC_2.0) [SUSv3]	lchown(GLIBC_2.0) [SUSv3]
link(GLIBC_2.0) [LSB]	lockf(GLIBC_2.0) [SUSv3]	lseek(GLIBC_2.0) [SUSv3]	mkdir(GLIBC_2.0) [SUSv3]
mkfifo(GLIBC_2.0) [SUSv3]	mlock(GLIBC_2.0) [SUSv3]	mlockall(GLIBC_2.0) [SUSv3]	mmap(GLIBC_2.0) [SUSv3]
mprotect(GLIBC_2.0) [SUSv3]	msync(GLIBC_2.0) [SUSv3]	munlock(GLIBC_2.0) [SUSv3]	munlockall(GLIBC_2.0) [SUSv3]
munmap(GLIBC_2.0) [SUSv3]	nanosleep(GLIBC_2.0) [SUSv3]	nice(GLIBC_2.0) [SUSv3]	open(GLIBC_2.0) [SUSv3]
opendir(GLIBC_2.0) [SUSv3]	pathconf(GLIBC_2.0) [SUSv3]	pause(GLIBC_2.0) [SUSv3]	pipe(GLIBC_2.0) [SUSv3]
poll(GLIBC_2.0) [SUSv3]	read(GLIBC_2.0) [SUSv3]	readdir(GLIBC_2.0) [SUSv3]	readdir_r(GLIBC_2.0) [SUSv3]
readlink(GLIBC_2.0) [SUSv3]	readv(GLIBC_2.0) [SUSv3]	rename(GLIBC_2.0) [SUSv3]	rmdir(GLIBC_2.0) [SUSv3]
sbrk(GLIBC_2.0) [SUSv2]	sched_get_priority_max(GLIBC_2.0) [SUSv3]	sched_get_priority_min(GLIBC_2.0) [SUSv3]	sched_getparam(GLIBC_2.0) [SUSv3]
sched_getscheduler(GLIBC_2.0) [SUSv3]	sched_rr_get_interval(GLIBC_2.0) [SUSv3]	sched_setparam(GLIBC_2.0) [SUSv3]	sched_setscheduler(GLIBC_2.0) [SUSv3]
sched_yield(GLIBC_2.0) [SUSv3]	select(GLIBC_2.0) [SUSv3]	setcontext(GLIBC_2.0) [SUSv3]	setegid(GLIBC_2.0) [SUSv3]
seteuid(GLIBC_2.0) [SUSv3]	setgid(GLIBC_2.0) [SUSv3]	setitimer(GLIBC_2.0) [SUSv3]	setpgid(GLIBC_2.0) [SUSv3]
setpgrp(GLIBC_2.0) [SUSv3]	setpriority(GLIBC_2.0) [SUSv3]	setregid(GLIBC_2.0) [SUSv3]	setreuid(GLIBC_2.0) [SUSv3]
setrlimit(GLIBC_2.0) [SUSv3]	setrlimit64(GLIBC_2.1) [LFS]	setsid(GLIBC_2.0) [SUSv3]	setuid(GLIBC_2.0) [SUSv3]
sleep(GLIBC_2.0) [SUSv3]	statvfs(GLIBC_2.1) [SUSv3]	stime(GLIBC_2.0) [LSB]	symlink(GLIBC_2.0) [SUSv3]
sync(GLIBC_2.0) [SUSv3]	sysconf(GLIBC_2.0) [SUSv3]	time(GLIBC_2.0) [SUSv3]	times(GLIBC_2.0) [SUSv3]
truncate(GLIBC_2.0) [SUSv3]	ulimit(GLIBC_2.0) [SUSv3]	umask(GLIBC_2.0) [SUSv3]	uname(GLIBC_2.0) [SUSv3]
unlink(GLIBC_2.0) [LSB]	utime(GLIBC_2.0) [SUSv3]	utimes(GLIBC_2.0) [SUSv3]	vfork(GLIBC_2.0) [SUSv3]
wait(GLIBC_2.0) [SUSv3]	wait4(GLIBC_2.0) [LSB]	waitpid(GLIBC_2.0) [LSB]	write(GLIBC_2.0) [SUSv3]

36

writev(GLIBC_2.0))[SUSv3]			
-------------------------------	--	--	--

11.2.3 Standard I/O

37

11.2.3.1 Interfaces for Standard I/O

38

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.

39

40

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

_IO_feof(GLIBC_2.0)[1]	fgetpos(GLIBC_2.2)[2]	fsetpos(GLIBC_2.2)[2]	putchar(GLIBC_2.0)[2]	sscanf(GLIBC_2.0)[1]
_IO_getc(GLIBC_2.0)[1]	fgets(GLIBC_2.0)[2]	ftell(GLIBC_2.0)[2]	putchar_unlocked(GLIBC_2.0)[2]	telldir(GLIBC_2.0)[2]
_IO_putc(GLIBC_2.0)[1]	fgetwc_unlocked(GLIBC_2.2)[1]	ftello(GLIBC_2.1)[2]	puts(GLIBC_2.0)[2]	tempnam(GLIBC_2.0)[2]
_IO_puts(GLIBC_2.0)[1]	fileno(GLIBC_2.0)[2]	fwrite(GLIBC_2.0)[2]	putw(GLIBC_2.0)[3]	ungetc(GLIBC_2.0)[2]
asprintf(GLIBC_2.0)[1]	flockfile(GLIBC_2.0)[2]	getc(GLIBC_2.0)[2]	remove(GLIBC_2.0)[2]	vasprintf(GLIBC_2.0)[1]
clearerr(GLIBC_2.0)[2]	fopen(GLIBC_2.1)[2]	getc_unlocked(GLIBC_2.0)[2]	rewind(GLIBC_2.0)[2]	vdprintf(GLIBC_2.0)[1]
etermid(GLIBC_2.0)[2]	fprintf(GLIBC_2.0)[2]	getchar(GLIBC_2.0)[2]	rewinddir(GLIBC_2.0)[2]	vfprintf(GLIBC_2.0)[2]
fclose(GLIBC_2.1)[2]	fputc(GLIBC_2.0)[2]	getchar_unlocked(GLIBC_2.0)[2]	scanf(GLIBC_2.0)[1]	vprintf(GLIBC_2.0)[2]
fdopen(GLIBC_2.1)[2]	fputs(GLIBC_2.0)[2]	getw(GLIBC_2.0)[3]	seekdir(GLIBC_2.0)[2]	vsprintf(GLIBC_2.0)[2]
feof(GLIBC_2.0)[2]	fread(GLIBC_2.0)[2]	pclose(GLIBC_2.1)[2]	setbuf(GLIBC_2.0)[2]	vsprintf(GLIBC_2.0)[2]
ferror(GLIBC_2.0)[2]	freopen(GLIBC_2.0)[2]	popen(GLIBC_2.1)[2]	setbuffer(GLIBC_2.0)[1]	
fflush(GLIBC_2.0)[2]	fscanf(GLIBC_2.0)[1]	printf(GLIBC_2.0)[2]	setvbuf(GLIBC_2.0)[2]	
fflush_unlocked(GLIBC_2.0)[1]	fseek(GLIBC_2.0)[2]	putc(GLIBC_2.0)[2]	snprintf(GLIBC_2.0)[2]	
fgetc(GLIBC_2.0)[2]	fseeko(GLIBC_2.1)[2]	putc_unlocked(GLIBC_2.0)[2]	sprintf(GLIBC_2.0)[2]	

42

		[2]		
--	--	-----	--	--

43

Referenced Specification(s)

44

[H]

_IO_feof(GLIBC_2.0) [LSB]	_IO_getc(GLIBC_2.0) [LSB]	_IO_putc(GLIBC_2.0) [LSB]	_IO_puts(GLIBC_2.0) [LSB]
asprintf(GLIBC_2.0) [LSB]	clearerr(GLIBC_2.0) [SUSv3]	ctermid(GLIBC_2.0) [SUSv3]	fclose(GLIBC_2.1) [SUSv3]
fdopen(GLIBC_2.1) [SUSv3]	feof(GLIBC_2.0) [SUSv3]	ferror(GLIBC_2.0) [SUSv3]	fflush(GLIBC_2.0) [SUSv3]
fflush_unlocked(GLIBC_2.0) [LSB]	fgetc(GLIBC_2.0) [SUSv3]	fgetpos(GLIBC_2.2) [SUSv3]	fgets(GLIBC_2.0) [SUSv3]
fgetwc_unlocked(GLIBC_2.2) [LSB]	fileno(GLIBC_2.0) [SUSv3]	flockfile(GLIBC_2.0) [SUSv3]	fopen(GLIBC_2.1) [SUSv3]
fprintf(GLIBC_2.0) [SUSv3]	fputc(GLIBC_2.0) [SUSv3]	fputs(GLIBC_2.0) [SUSv3]	fread(GLIBC_2.0) [SUSv3]
freopen(GLIBC_2.0) [SUSv3]	fscanf(GLIBC_2.0) [LSB]	fseek(GLIBC_2.0) [SUSv3]	fseeko(GLIBC_2.1) [SUSv3]
fsetpos(GLIBC_2.2) [SUSv3]	ftell(GLIBC_2.0) [SUSv3]	ftello(GLIBC_2.1) [SUSv3]	fwrite(GLIBC_2.0) [SUSv3]
getc(GLIBC_2.0) [SUSv3]	getc_unlocked(GLIBC_2.0) [SUSv3]	getchar(GLIBC_2.0) [SUSv3]	getchar_unlocked(GLIBC_2.0) [SUSv3]
getw(GLIBC_2.0) [SUSv2]	pclose(GLIBC_2.1) [SUSv3]	popen(GLIBC_2.1) [SUSv3]	printf(GLIBC_2.0) [SUSv3]
putc(GLIBC_2.0) [SUSv3]	putc_unlocked(GLIBC_2.0) [SUSv3]	putchar(GLIBC_2.0) [SUSv3]	putchar_unlocked(GLIBC_2.0) [SUSv3]
puts(GLIBC_2.0) [SUSv3]	putw(GLIBC_2.0) [SUSv2]	remove(GLIBC_2.0) [SUSv3]	rewind(GLIBC_2.0) [SUSv3]
rewinddir(GLIBC_2.0) [SUSv3]	scanf(GLIBC_2.0) [LSB]	seekdir(GLIBC_2.0) [SUSv3]	setbuf(GLIBC_2.0) [SUSv3]
setbuffer(GLIBC_2.0) [LSB]	setvbuf(GLIBC_2.0) [SUSv3]	snprintf(GLIBC_2.0) [SUSv3]	sprintf(GLIBC_2.0) [SUSv3]
sscanf(GLIBC_2.0) [LSB]	telldir(GLIBC_2.0) [SUSv3]	tempnam(GLIBC_2.0) [SUSv3]	ungetc(GLIBC_2.0) [SUSv3]
vasprintf(GLIBC_2.0) [LSB]	vdprintf(GLIBC_2.0) [LSB]	vfprintf(GLIBC_2.0) [SUSv3]	vprintf(GLIBC_2.0) [SUSv3]
vsnprintf(GLIBC_2.0) [SUSv3]	vsprintf(GLIBC_2.0) [SUSv3]		

45

46

47

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

48

[2]. ISO POSIX (2003)

49

[3]. SUSv2

50

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.

52

53

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

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

54

Referenced Specification(s)

55

[1]. ISO POSIX (2003)

56

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

57

11.2.4 Signal Handling

58

11.2.4.1 Interfaces for Signal Handling

59

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.

60

61

Table 11-6 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.1) [1]	sigaction(GLIBC_2.0) [2]	sighold(GLIBC_2.1) [2]	sigorset(GLIBC_2.0) [1]	sigset(GLIBC_2.1) [2]
__libc_current_sigrtmin(GLIBC_2.1) [1]	sigaddset(GLIBC_2.0) [2]	sigignore(GLIBC_2.1) [2]	sigpause(GLIBC_2.0) [2]	sigsuspend(GLIBC_2.0) [2]
__sigsetjmp(GLIBC_2.0) [1]	sigaltstack(GLIBC_2.0) [2]	siginterrupt(GLIBC_2.0) [2]	sigpending(GLIBC_2.0) [2]	sigtimedwait(GLIBC_2.1) [2]
__sysv_signal(GLIBC_2.0) [1]	sigandset(GLIBC_2.0) [1]	sigisemptyset(GLIBC_2.0) [1]	sigprocmask(GLIBC_2.0) [2]	sigwait(GLIBC_2.0) [2]
bsd_signal(GLIBC_2.0) [2]	sigdelset(GLIBC_2.0) [2]	sigismember(GLIBC_2.0) [2]	sigqueue(GLIBC_2.1) [2]	sigwaitinfo(GLIBC_2.1) [2]
psignal(GLIBC_2.0) [1]	sigemptyset(GLIBC_2.0) [2]	siglongjmp(GLIBC_2.0) [2]	sigrelse(GLIBC_2.1) [2]	
raise(GLIBC_2.0) [2]	sigfillset(GLIBC_2.0) [2]	signal(GLIBC_2.0) [2]	sigreturn(GLIBC_2.0) [1]	

63

Referenced Specification(s)

64

65

~~[1]~~

__libc_current_sigrtmax(GLIBC_2.1) [LSB]	__libc_current_sigrtmin(GLIBC_2.1) [LSB]	__sigsetjmp(GLIBC_2.0) [LSB]	__sysv_signal(GLIBC_2.0) [LSB]
bsd_signal(GLIBC_2.0) [SUSv3]	psignal(GLIBC_2.0) [LSB]	raise(GLIBC_2.0) [SUSv3]	sigaction(GLIBC_2.0) [SUSv3]
sigaddset(GLIBC_2.0) [SUSv3]	sigaltstack(GLIBC_2.0) [SUSv3]	sigandset(GLIBC_2.0) [LSB]	sigdelset(GLIBC_2.0) [SUSv3]
sigemptyset(GLIBC_2.0) [SUSv3]	sigfillset(GLIBC_2.0) [SUSv3]	sighold(GLIBC_2.1) [SUSv3]	sigignore(GLIBC_2.1) [SUSv3]
siginterrupt(GLIBC_2.0) [SUSv3]	sigisemptyset(GLIBC_2.0) [LSB]	sigismember(GLIBC_2.0) [SUSv3]	siglongjmp(GLIBC_2.0) [SUSv3]
signal(GLIBC_2.0) [SUSv3]	sigorset(GLIBC_2.0) [LSB]	sigpause(GLIBC_2.0) [SUSv3]	sigpending(GLIBC_2.0) [SUSv3]
sigprocmask(GLIBC_2.0) [SUSv3]	sigqueue(GLIBC_2.1) [SUSv3]	sigrelse(GLIBC_2.1) [SUSv3]	sigreturn(GLIBC_2.0) [LSB]
sigset(GLIBC_2.1) [SUSv3]	sigsuspend(GLIBC_2.0) [SUSv3]	sigtimedwait(GLIBC_2.1) [SUSv3]	sigwait(GLIBC_2.0) [SUSv3]
sigwaitinfo(GLIBC_2.1) [SUSv3]			

66

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

67

68

69

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

70

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.

71

72

73

Table 11-7 libc - Signal Handling Data Interfaces

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

74

75

Referenced Specification(s)

76

~~[1]. this specification~~

77

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

11.2.5 Localization Functions

78

11.2.5.1 Interfaces for Localization Functions

79

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.

80

81

82

Table 11-8 libc - Localization Functions Function Interfaces

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

83

84

Referenced Specification(s)

85

~~[1].~~

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

86

87

88

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

89

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

90

91

92

~~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.~~

93

Table 11-9 libc - Localization Functions Data Interfaces

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

94

95

Referenced Specification(s)

96

~~[1]. this specification~~

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

97

11.2.6 Socket Interface

11.2.6.1 Interfaces for Socket Interface

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

Table 11-10 libc - Socket Interface Function Interfaces

<code>__h_errno_location(GLIBC_2.0)</code> [1]	<code>gethostname(GLIBC_2.0)</code> [2]	<code>if_nameindex(GLIBC_2.1)</code> [2]	<code>send(GLIBC_2.0)</code> [2]	<code>socket(GLIBC_2.0)</code> [2]
<code>accept(GLIBC_2.0)</code> [2]	<code>getpeername(GLIBC_2.0)</code> [2]	<code>if_nametoindex(GLIBC_2.1)</code> [2]	<code>sendmsg(GLIBC_2.0)</code> [2]	<code>socketpair(GLIBC_2.0)</code> [2]
<code>bind(GLIBC_2.0)</code> [2]	<code>getsockname(GLIBC_2.0)</code> [2]	<code>listen(GLIBC_2.0)</code> [2]	<code>sendto(GLIBC_2.0)</code> [2]	
<code>bindresvport(GLIBC_2.0)</code> [1]	<code>getsockopt(GLIBC_2.0)</code> [1]	<code>recv(GLIBC_2.0)</code> [2]	<code>setsockopt(GLIBC_2.0)</code> [1]	
<code>connect(GLIBC_2.0)</code> [2]	<code>if_freenameindex(GLIBC_2.1)</code> [2]	<code>recvfrom(GLIBC_2.0)</code> [2]	<code>shutdown(GLIBC_2.0)</code> [2]	
<code>gethostid(GLIBC_2.0)</code> [2]	<code>if_indextoname(GLIBC_2.1)</code> [2]	<code>recvmsg(GLIBC_2.0)</code> [2]	<code>socketatmark(GLIBC_2.2.4)</code> [2]	
<code>__h_errno_location(GLIBC_2.0)</code> [LSB]	<code>accept(GLIBC_2.0)</code> [SUSv3]	<code>bind(GLIBC_2.0)</code> [SUSv3]	<code>bindresvport(GLIBC_2.0)</code> [LSB]	
<code>connect(GLIBC_2.0)</code> [SUSv3]	<code>gethostid(GLIBC_2.0)</code> [SUSv3]	<code>gethostname(GLIBC_2.0)</code> [SUSv3]	<code>getpeername(GLIBC_2.0)</code> [SUSv3]	
<code>getsockname(GLIBC_2.0)</code> [SUSv3]	<code>getsockopt(GLIBC_2.0)</code> [LSB]	<code>if_freenameindex(GLIBC_2.1)</code> [SUSv3]	<code>if_indextoname(GLIBC_2.1)</code> [SUSv3]	
<code>if_nameindex(GLIBC_2.1)</code> [SUSv3]	<code>if_nametoindex(GLIBC_2.1)</code> [SUSv3]	<code>listen(GLIBC_2.0)</code> [SUSv3]	<code>recv(GLIBC_2.0)</code> [SUSv3]	
<code>recvfrom(GLIBC_2.0)</code> [SUSv3]	<code>recvmsg(GLIBC_2.0)</code> [SUSv3]	<code>send(GLIBC_2.0)</code> [SUSv3]	<code>sendmsg(GLIBC_2.0)</code> [SUSv3]	
<code>sendto(GLIBC_2.0)</code> [SUSv3]	<code>setsockopt(GLIBC_2.0)</code> [LSB]	<code>shutdown(GLIBC_2.0)</code> [SUSv3]	<code>socketatmark(GLIBC_2.2.4)</code> [SUSv3]	
<code>socket(GLIBC_2.0)</code> [SUSv3]	<code>socketpair(GLIBC_2.0)</code> [SUSv3]			

Referenced Specification(s)

105 ~~[1]. this specification~~

106 ~~[2]. ISO POSIX (2003)~~

11.2.7 Wide Characters

11.2.7.1 Interfaces for Wide Characters

107
108 An LSB conforming implementation shall provide the architecture specific functions
109 for Wide Characters specified in Table 11-11, with the full mandatory functionality
110 as described in the referenced underlying specification.

111 **Table 11-11 libc - Wide Characters Function Interfaces**

__westod_int ernal(GLIBC_ 2.0) [1]	mbsinit(GLIB C_2.0) [2]	vwscanf(GLIB C_2.2) [1]	wesnlen(GLIB C_2.1) [1]	westoumax(G LIBC_2.1) [2]
__westof_inte rnal(GLIBC_2 .0) [1]	mbsrtowes(GLIBC_2.0) [1]	wcpepy(GLIB C_2.0) [1]	wesnrtoombs(GLIBC_2.0) [1]	westouq(GLI BC_2.0) [1]
__westol_inte rnal(GLIBC_2 .0) [1]	mbsrtowes(G LIBC_2.0) [2]	wepnepy(GLI BC_2.0) [1]	wespbrk(GLI BC_2.0) [2]	weswes(GLIB C_2.1) [2]
__westold_int ernal(GLIBC_ 2.0) [1]	mbstowes(GL IBC_2.0) [2]	wertomb(GLI BC_2.0) [2]	wesrchr(GLIB C_2.0) [2]	weswidth(GL IBC_2.0) [2]
__westoul_int ernal(GLIBC_ 2.0) [1]	mbtowc(GLIB C_2.0) [2]	wescasecmp(GLIBC_2.1) [1]	wesrtombs(G LIBC_2.0) [2]	wesxfrm(GLI BC_2.0) [2]
btowc(GLIBC _2.0) [2]	putwc(GLIBC _2.2) [2]	wecat(GLIBC _2.0) [2]	wesspn(GLIB C_2.0) [2]	wetob(GLIBC _2.0) [2]
fgetwc(GLIBC _2.2) [2]	putwchar(GLI BC_2.2) [2]	weschr(GLIB C_2.0) [2]	wesstr(GLIBC _2.0) [2]	wetomb(GLIB C_2.0) [2]
fgetwsw(GLIBC _2.2) [2]	swprintf(GLI BC_2.2) [2]	wescmp(GLIB C_2.0) [2]	westod(GLIB C_2.0) [2]	wetrans(GLIB C_2.0) [2]
fputwc(GLIB C_2.2) [2]	swscanf(GLIB C_2.2) [1]	wescoll(GLIB C_2.0) [2]	westof(GLIBC _2.0) [2]	wetype(GLIB C_2.0) [2]
fputwsw(GLIB C_2.2) [2]	towctrans(GL IBC_2.0) [2]	wescpy(GLIB C_2.0) [2]	westoimax(G LIBC_2.1) [2]	wewidth(GLI BC_2.0) [2]
fwide(GLIBC _2.2) [2]	towlower(GLI BC_2.0) [2]	wespsn(GLI BC_2.0) [2]	westok(GLIB C_2.0) [2]	wmemchr(GL IBC_2.0) [2]
fwprintf(GLI BC_2.2) [2]	towupper(GL IBC_2.0) [2]	wesdup(GLIB C_2.0) [1]	westol(GLIBC _2.0) [2]	wmememp(G LIBC_2.0) [2]
fwscanf(GLIB C_2.2) [1]	ungetwc(GLI BC_2.2) [2]	wesftime(GLI BC_2.2) [2]	westold(GLIB C_2.0) [2]	wmemepy(G LIBC_2.0) [2]
getwc(GLIBC _2.2) [2]	vfwprintf(GLI BC_2.2) [2]	weslen(GLIB C_2.0) [2]	westoll(GLIB C_2.1) [2]	wmemmove(GLIBC_2.0)

				[2]
getwchar(GLIBC_2.2) [2]	vfwscanf(GLIBC_2.2) [1]	wcscasecmp (GLIBC_2.1) [1]	wcstombs(GLIBC_2.0) [2]	wmemset(GLIBC_2.0) [2]
mblen(GLIBC_2.0) [2]	vswprintf(GLIBC_2.2) [2]	wcscat(GLIBC_2.0) [2]	wcstoq(GLIBC_2.0) [1]	wprintf(GLIBC_2.2) [2]
mbrlen(GLIBC_2.0) [2]	vswscanf(GLIBC_2.2) [1]	wcsncmp(GLIBC_2.0) [2]	wcstoul(GLIBC_2.0) [2]	wscanf(GLIBC_2.2) [1]
mbrtowc(GLIBC_2.0) [2]	vwprintf(GLIBC_2.2) [2]	wcsncpy(GLIBC_2.0) [2]	wcstoull(GLIBC_2.1) [2]	

112

113

Referenced Specification(s)

114

[1]. this specification

115

[2]. ISO POSIX (2003)

__wcstod_internal (GLIBC_2.0) [LSB]	__wcstof_internal (GLIBC_2.0) [LSB]	__wcstol_internal (GLIBC_2.0) [LSB]	__wcstold_internal (GLIBC_2.0) [LSB]
__wcstoul_internal (GLIBC_2.0) [LSB]	btowc (GLIBC_2.0) [SUSv3]	fgetwc (GLIBC_2.2) [SUSv3]	fgetws (GLIBC_2.2) [SUSv3]
fputwc (GLIBC_2.2) [SUSv3]	fputws (GLIBC_2.2) [SUSv3]	fwide (GLIBC_2.2) [SUSv3]	fwprintf (GLIBC_2.2) [SUSv3]
fwscanf (GLIBC_2.2) [LSB]	getwc (GLIBC_2.2) [SUSv3]	getwchar (GLIBC_2.2) [SUSv3]	mblen (GLIBC_2.0) [SUSv3]
mbrlen (GLIBC_2.0) [SUSv3]	mbrtowc (GLIBC_2.0) [SUSv3]	mbsinit (GLIBC_2.0) [SUSv3]	mbsrtowcs (GLIBC_2.0) [LSB]
mbsrtowcs (GLIBC_2.0) [SUSv3]	mbstowcs (GLIBC_2.0) [SUSv3]	mbtowc (GLIBC_2.0) [SUSv3]	putwc (GLIBC_2.2) [SUSv3]
putwchar (GLIBC_2.2) [SUSv3]	swprintf (GLIBC_2.2) [SUSv3]	swscanf (GLIBC_2.2) [LSB]	towctrans (GLIBC_2.0) [SUSv3]
tolower (GLIBC_2.0) [SUSv3]	toupper (GLIBC_2.0) [SUSv3]	ungetwc (GLIBC_2.2) [SUSv3]	vfwprintf (GLIBC_2.2) [SUSv3]
vfwscanf (GLIBC_2.2) [LSB]	vswprintf (GLIBC_2.2) [SUSv3]	vswscanf (GLIBC_2.2) [LSB]	vwprintf (GLIBC_2.2) [SUSv3]
vwscanf (GLIBC_2.2) [LSB]	wcpcpy (GLIBC_2.0) [LSB]	wcpncpy (GLIBC_2.0) [LSB]	wcrtomb (GLIBC_2.0) [SUSv3]
wcscasecmp (GLIBC_2.1) [LSB]	wcscat (GLIBC_2.0) [SUSv3]	wcschr (GLIBC_2.0) [SUSv3]	wcscmp (GLIBC_2.0) [SUSv3]
wscoll (GLIBC_2.0) [SUSv3]	wcscpy (GLIBC_2.0) [SUSv3]	wcscspn (GLIBC_2.0) [SUSv3]	wcsdup (GLIBC_2.0) [LSB]
wcsftime (GLIBC_2.2) [SUSv3]	wcslen (GLIBC_2.0) [SUSv3]	wcscasecmp (GLIBC_2.1) [LSB]	wcscat (GLIBC_2.0) [SUSv3]

wcsncmp(GLIBC_2.0) [SUSv3]	wcsncpy(GLIBC_2.0) [SUSv3]	wcsnlen(GLIBC_2.1) [LSB]	wcsnrtombs(GLIBC_2.0) [LSB]
wcspbrk(GLIBC_2.0) [SUSv3]	wcsrchr(GLIBC_2.0) [SUSv3]	wcsrtombs(GLIBC_2.0) [SUSv3]	wcsspn(GLIBC_2.0) [SUSv3]
wcsstr(GLIBC_2.0) [SUSv3]	wctod(GLIBC_2.0) [SUSv3]	wcstof(GLIBC_2.0) [SUSv3]	wcstoimax(GLIBC_2.1) [SUSv3]
wctok(GLIBC_2.0) [SUSv3]	wctol(GLIBC_2.0) [SUSv3]	wctold(GLIBC_2.0) [SUSv3]	wctoll(GLIBC_2.1) [SUSv3]
wctombs(GLIBC_2.0) [SUSv3]	wctoq(GLIBC_2.0) [LSB]	wctoul(GLIBC_2.0) [SUSv3]	wctoull(GLIBC_2.1) [SUSv3]
wctoumax(GLIBC_2.1) [SUSv3]	wctouq(GLIBC_2.0) [LSB]	wcswcs(GLIBC_2.1) [SUSv3]	wcswidth(GLIBC_2.0) [SUSv3]
wcsxfrm(GLIBC_2.0) [SUSv3]	wctob(GLIBC_2.0) [SUSv3]	wctomb(GLIBC_2.0) [SUSv3]	wctrans(GLIBC_2.0) [SUSv3]
wctype(GLIBC_2.0) [SUSv3]	wcwidth(GLIBC_2.0) [SUSv3]	wmemchr(GLIBC_2.0) [SUSv3]	wmemcmp(GLIBC_2.0) [SUSv3]
wmemcpy(GLIBC_2.0) [SUSv3]	wmemmove(GLIBC_2.0) [SUSv3]	wmemset(GLIBC_2.0) [SUSv3]	wprintf(GLIBC_2.2) [SUSv3]
wscanf(GLIBC_2.2) [LSB]			

116

11.2.8 String Functions

117

11.2.8.1 Interfaces for String Functions

118

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.

119

120

121

Table 11-12 libc - String Functions Function Interfaces

<code>__memcpy(GLIBC_2.0)</code> [1]	<code>bzero(GLIBC_2.0)</code> [2]	<code>stresestr(GLIBC_2.1)</code> [1]	<code>strncat(GLIBC_2.0)</code> [2]	<code>strtok(GLIBC_2.0)</code> [2]
<code>__rawmemchr(GLIBC_2.1)</code> [1]	<code>ffs(GLIBC_2.0)</code> [2]	<code>streat(GLIBC_2.0)</code> [2]	<code>strncmp(GLIBC_2.0)</code> [2]	<code>strtok_r(GLIBC_2.0)</code> [2]
<code>__stpncpy(GLIBC_2.0)</code> [1]	<code>index(GLIBC_2.0)</code> [2]	<code>strchr(GLIBC_2.0)</code> [2]	<code>strncpy(GLIBC_2.0)</code> [2]	<code>strtol(GLIBC_2.0)</code> [2]
<code>__strdup(GLIBC_2.0)</code> [1]	<code>memcpy(GLIBC_2.0)</code> [2]	<code>strcmp(GLIBC_2.0)</code> [2]	<code>strndup(GLIBC_2.0)</code> [1]	<code>strtoll(GLIBC_2.0)</code> [2]
<code>__strtod_internal(GLIBC_2.0)</code> [1]	<code>memchr(GLIBC_2.0)</code> [2]	<code>streq(GLIBC_2.0)</code> [2]	<code>strlen(GLIBC_2.0)</code> [1]	<code>strtoq(GLIBC_2.0)</code> [1]
<code>__strtof_internal(GLIBC_2.0)</code> [1]	<code>memcmp(GLIBC_2.0)</code> [2]	<code>strcpy(GLIBC_2.0)</code> [2]	<code>strpbrk(GLIBC_2.0)</code> [2]	<code>strtoull(GLIBC_2.0)</code> [2]

<code>nal(GLIBC_2.0)</code> [1]	<code>BC_2.0)</code> [2]	<code>_2.0)</code> [2]	<code>C_2.0)</code> [2]	<code>C_2.0)</code> [2]
<code>__strtok_r(GLIBC_2.0)</code> [1]	<code>memcpy(GLIBC_2.0)</code> [2]	<code>strespn(GLIBC_2.0)</code> [2]	<code>strptime(GLIBC_2.0)</code> [1]	<code>strtoumax(GLIBC_2.1)</code> [2]
<code>__strtol_internal(GLIBC_2.0)</code> [1]	<code>memmove(GLIBC_2.0)</code> [2]	<code>strdup(GLIBC_2.0)</code> [2]	<code>strchr(GLIBC_2.0)</code> [2]	<code>strtouq(GLIBC_2.0)</code> [1]
<code>__strtol_internal(GLIBC_2.0)</code> [1]	<code>memrchr(GLIBC_2.2)</code> [1]	<code>strerror(GLIBC_2.0)</code> [2]	<code>strsep(GLIBC_2.0)</code> [1]	<code>strxfrm(GLIBC_2.0)</code> [2]
<code>__strtoll_internal(GLIBC_2.0)</code> [1]	<code>memset(GLIBC_2.0)</code> [2]	<code>strerror_r(GLIBC_2.0)</code> [1]	<code>strsignal(GLIBC_2.0)</code> [1]	<code>swab(GLIBC_2.0)</code> [2]
<code>__strtoul_internal(GLIBC_2.0)</code> [1]	<code>rindex(GLIBC_2.0)</code> [2]	<code>strfmon(GLIBC_2.0)</code> [2]	<code>strspn(GLIBC_2.0)</code> [2]	
<code>__strtoull_internal(GLIBC_2.0)</code> [1]	<code>stpcpy(GLIBC_2.0)</code> [1]	<code>strtime(GLIBC_2.0)</code> [2]	<code>strstr(GLIBC_2.0)</code> [2]	
<code>bcmp(GLIBC_2.0)</code> [2]	<code>stpncpy(GLIBC_2.0)</code> [1]	<code>strlen(GLIBC_2.0)</code> [2]	<code>strtof(GLIBC_2.0)</code> [2]	
<code>bcopy(GLIBC_2.0)</code> [2]	<code>strcasecmp(GLIBC_2.0)</code> [2]	<code>strncasecmp(GLIBC_2.0)</code> [2]	<code>strtoimax(GLIBC_2.1)</code> [2]	
<code>__memcpy(GLIBC_2.0)</code> [LSB]	<code>__rawmemchr(GLIBC_2.1)</code> [LSB]	<code>__stpcpy(GLIBC_2.0)</code> [LSB]	<code>__strdup(GLIBC_2.0)</code> [LSB]	
<code>__strtod_internal(GLIBC_2.0)</code> [LSB]	<code>__strtof_internal(GLIBC_2.0)</code> [LSB]	<code>__strtok_r(GLIBC_2.0)</code> [LSB]	<code>__strtol_internal(GLIBC_2.0)</code> [LSB]	
<code>__strtol_internal(GLIBC_2.0)</code> [LSB]	<code>__strtoll_internal(GLIBC_2.0)</code> [LSB]	<code>__strtoul_internal(GLIBC_2.0)</code> [LSB]	<code>__strtoull_internal(GLIBC_2.0)</code> [LSB]	
<code>bcmp(GLIBC_2.0)</code> [SUSv3]	<code>bcopy(GLIBC_2.0)</code> [SUSv3]	<code>bzero(GLIBC_2.0)</code> [SUSv3]	<code>ffs(GLIBC_2.0)</code> [SUSv3]	
<code>index(GLIBC_2.0)</code> [SUSv3]	<code>memccpy(GLIBC_2.0)</code> [SUSv3]	<code>memchr(GLIBC_2.0)</code> [SUSv3]	<code>memcmp(GLIBC_2.0)</code> [SUSv3]	
<code>memcpy(GLIBC_2.0)</code> [SUSv3]	<code>memmove(GLIBC_2.0)</code> [SUSv3]	<code>memrchr(GLIBC_2.2)</code> [LSB]	<code>memset(GLIBC_2.0)</code> [SUSv3]	
<code>rindex(GLIBC_2.0)</code> [SUSv3]	<code>stpcpy(GLIBC_2.0)</code> [LSB]	<code>stpncpy(GLIBC_2.0)</code> [LSB]	<code>strcasecmp(GLIBC_2.0)</code> [SUSv3]	
<code>strcasestr(GLIBC_2.1)</code> [LSB]	<code>strcat(GLIBC_2.0)</code> [SUSv3]	<code>strchr(GLIBC_2.0)</code> [SUSv3]	<code>strcmp(GLIBC_2.0)</code> [SUSv3]	
<code>strcoll(GLIBC_2.0)</code> [SUSv3]	<code>strcpy(GLIBC_2.0)</code> [SUSv3]	<code>strcspn(GLIBC_2.0)</code> [SUSv3]	<code>strdup(GLIBC_2.0)</code> [SUSv3]	

strerror(GLIBC_2.0) [SUSv3]	strerror_r(GLIBC_2.0) [LSB]	strfmon(GLIBC_2.0) [SUSv3]	strftime(GLIBC_2.0) [SUSv3]
strlen(GLIBC_2.0) [SUSv3]	strncasecmp(GLIBC_2.0) [SUSv3]	strncat(GLIBC_2.0) [SUSv3]	strncmp(GLIBC_2.0) [SUSv3]
strncpy(GLIBC_2.0) [SUSv3]	strndup(GLIBC_2.0) [LSB]	strnlen(GLIBC_2.0) [LSB]	strpbrk(GLIBC_2.0) [SUSv3]
strptime(GLIBC_2.0) [LSB]	strchr(GLIBC_2.0) [SUSv3]	strsep(GLIBC_2.0) [LSB]	strsignal(GLIBC_2.0) [LSB]
strspn(GLIBC_2.0) [SUSv3]	strstr(GLIBC_2.0) [SUSv3]	strtof(GLIBC_2.0) [SUSv3]	strtoimax(GLIBC_2.1) [SUSv3]
strtok(GLIBC_2.0) [SUSv3]	strtok_r(GLIBC_2.0) [SUSv3]	strtold(GLIBC_2.0) [SUSv3]	strtoll(GLIBC_2.0) [SUSv3]
strtoq(GLIBC_2.0) [LSB]	strtoull(GLIBC_2.0) [SUSv3]	strtoumax(GLIBC_2.1) [SUSv3]	strtouq(GLIBC_2.0) [LSB]
strxfrm(GLIBC_2.0) [SUSv3]	swab(GLIBC_2.0) [SUSv3]		

122

123

Referenced Specification(s)

124

~~[1]. this specification~~

125

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

11.2.9 IPC Functions

126

11.2.9.1 Interfaces for IPC Functions

127

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.

128

129

130

Table 11-13 libc - IPC Functions Function Interfaces

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

131

132

Referenced Specification(s)

133

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

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

[SUSv3]) [SUSv3]) [SUSv3]	0) [SUSv3]
---------	-----------	-----------	------------

11.2.10 Regular Expressions

11.2.10.1 Interfaces for Regular Expressions

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

Table 11-14 libc - Regular Expressions Function Interfaces

regcomp(GLIBC_2.0) [1]	regerror(GLIBC_2.0) [1]	regexexec(GLIBC_2.3.4) [2]	regfree(GLIBC_2.0) [1]	
------------------------	-------------------------	----------------------------	------------------------	--

Referenced Specification(s)

[1]- ISO POSIX (2003)

[2]- this specification

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

11.2.11 Character Type Functions

11.2.11.1 Interfaces for Character Type Functions

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

Table 11-15 libc - Character Type Functions Function Interfaces

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

Referenced Specification(s)

152

~~[1]. this specification~~

153

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

__ctype_get_mb_c ur_max(GLIBC_2. 0) [LSB]	_tolower(GLIBC_ 2.0) [SUSv3]	_toupper(GLIBC_ 2.0) [SUSv3]	isalnum(GLIBC_2. 0) [SUSv3]
isalpha(GLIBC_2. 0) [SUSv3]	isascii(GLIBC_2.0) [SUSv3]	iscntrl(GLIBC_2.0) [SUSv3]	isdigit(GLIBC_2.0) [SUSv3]
isgraph(GLIBC_2. 0) [SUSv3]	islower(GLIBC_2. 0) [SUSv3]	isprint(GLIBC_2.0) [SUSv3]	ispunct(GLIBC_2. 0) [SUSv3]
isspace(GLIBC_2. 0) [SUSv3]	isupper(GLIBC_2. 0) [SUSv3]	iswalnum(GLIBC_ 2.0) [SUSv3]	iswalpha(GLIBC_ 2.0) [SUSv3]
iswblank(GLIBC_ 2.1) [SUSv3]	iswcntrl(GLIBC_2. .0) [SUSv3]	iswctype(GLIBC_ 2.0) [SUSv3]	iswdigit(GLIBC_2. .0) [SUSv3]
iswgraph(GLIBC_ 2.0) [SUSv3]	iswlower(GLIBC_ 2.0) [SUSv3]	iswprint(GLIBC_2. .0) [SUSv3]	iswpunct(GLIBC_ 2.0) [SUSv3]
iswspace(GLIBC_ 2.0) [SUSv3]	iswupper(GLIBC_ 2.0) [SUSv3]	iswxdigit(GLIBC_ 2.0) [SUSv3]	isxdigit(GLIBC_2. 0) [SUSv3]
toascii(GLIBC_2.0) [SUSv3]	tolower(GLIBC_2. 0) [SUSv3]	toupper(GLIBC_2. 0) [SUSv3]	

154

11.2.12 Time Manipulation

155

11.2.12.1 Interfaces for Time Manipulation

156

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.

157

158

159

Table 11-16 libc - Time Manipulation Function Interfaces

adjtime(GLIB C_2.0) [1]	etime(GLIBC_ 2.0) [2]	gmtime(GLIB C_2.0) [2]	localtime_r(G LIBC_2.0) [2]	ualarm(GLIB C_2.0) [2]
asctime(GLIB C_2.0) [2]	etime_r(GLIB C_2.0) [2]	gmtime_r(GL IBC_2.0) [2]	mktime(GLIB C_2.0) [2]	
asctime_r(GLI BC_2.0) [2]	diffime(GLIB C_2.0) [2]	localtime(GLI BC_2.0) [2]	tzset(GLIBC_ 2.0) [2]	

160

161

Referenced Specification(s)

162

~~[1].~~

adjtime(GLIBC_2. 0) [LSB]	asctime(GLIBC_2. 0) [SUSv3]	asctime_r(GLIBC_ 2.0) [SUSv3]	ctime(GLIBC_2.0) [SUSv3]
ctime_r(GLIBC_2. 0) [SUSv3]	diffime(GLIBC_2. 0) [SUSv3]	gmtime(GLIBC_2. 0) [SUSv3]	gmtime_r(GLIBC_ 2.0) [SUSv3]
localtime(GLIBC_ 2.0) [SUSv3]	localtime_r(GLIB C_2.0) [SUSv3]	mktime(GLIBC_2. 0) [SUSv3]	tzset(GLIBC_2.0) [SUSv3]

ualarm(GLIBC_2.0) [SUSv3]			
---------------------------	--	--	--

163

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

164

165

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

166

~~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.~~

167

168

169

Table 11-17 libc - Time Manipulation Data Interfaces

170

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

171

Referenced Specification(s)

172

~~[1]. this specification~~

173

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

174

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

175

11.2.13 Terminal Interface Functions

176

11.2.13.1 Interfaces for Terminal Interface Functions

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

177

178

179

Table 11-18 libc - Terminal Interface Functions Function Interfaces

180

cfgetispeed(GLIBC_2.0) [1]	cfsetispeed(GLIBC_2.0) [1]	tcdrain(GLIBC_2.0) [1]	tegetattr(GLIBC_2.0) [1]	tesendbreak(GLIBC_2.0) [1]
cfgetospeed(GLIBC_2.0) [1]	cfsetospeed(GLIBC_2.0) [1]	tcflow(GLIBC_2.0) [1]	tegetpgrp(GLIBC_2.0) [1]	tesetattr(GLIBC_2.0) [1]
cfmakeraw(GLIBC_2.0) [2]	cfsetspeed(GLIBC_2.0) [2]	tcflush(GLIBC_2.0) [1]	tegetsid(GLIBC_2.1) [1]	tesetpgrp(GLIBC_2.0) [1]

181

Referenced Specification(s)

182

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

183

~~[2]. this specification~~

184

cfgetispeed(GLIB	cfgetospeed(GLIB	cfmakeraw(GLIB	cfsetispeed(GLIB
-----------------------------	-----------------------------	---------------------------	-----------------------------

C_2.0) [SUSv3]	C_2.0) [SUSv3]	C_2.0) [LSB]	C_2.0) [SUSv3]
cfsetospeed(GLIBC_2.0) [SUSv3]	cfsetospeed(GLIBC_2.0) [LSB]	tcdrain(GLIBC_2.0) [SUSv3]	tcflow(GLIBC_2.0) [SUSv3]
tcflush(GLIBC_2.0) [SUSv3]	tcgetattr(GLIBC_2.0) [SUSv3]	tcgetpgrp(GLIBC_2.0) [SUSv3]	tcgetsid(GLIBC_2.0) [SUSv3]
tcsendbreak(GLIBC_2.0) [SUSv3]	tcsetattr(GLIBC_2.0) [SUSv3]	tcsetpgrp(GLIBC_2.0) [SUSv3]	

185

11.2.14 System Database Interface

186

11.2.14.1 Interfaces for System Database Interface

187

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.

188

189

190

Table 11-19 libc - System Database Interface Function Interfaces

endgrent(GLIBC_2.0) [1]	getgrgid_r(GLIBC_2.1.2) [1]	getprotoent(GLIBC_2.0) [1]	getservent(GLIBC_2.0) [1]	setgroups(GLIBC_2.0) [2]
endprotoent(GLIBC_2.0) [1]	getgrnam(GLIBC_2.0) [1]	getpwent(GLIBC_2.0) [1]	getutent(GLIBC_2.0) [2]	setprotoent(GLIBC_2.0) [1]
endpwent(GLIBC_2.0) [1]	getgrnam_r(GLIBC_2.1.2) [1]	getpwnam(GLIBC_2.0) [1]	getutent_r(GLIBC_2.0) [2]	setpwent(GLIBC_2.0) [1]
endservent(GLIBC_2.0) [1]	getgrouplist(GLIBC_2.2.4) [2]	getpwnam_r(GLIBC_2.1.2) [1]	getutxent(GLIBC_2.1) [1]	setservent(GLIBC_2.0) [1]
endutent(GLIBC_2.0) [3]	gethostbyaddr(GLIBC_2.0) [1]	getpwuid(GLIBC_2.0) [1]	getutxid(GLIBC_2.1) [1]	setutent(GLIBC_2.0) [2]
endutxent(GLIBC_2.1) [1]	gethostbyname(GLIBC_2.0) [1]	getpwuid_r(GLIBC_2.1.2) [1]	getutxline(GLIBC_2.1) [1]	setutxent(GLIBC_2.1) [1]
getgrent(GLIBC_2.0) [1]	getprotobynam(GLIBC_2.0) [1]	getservbyname(GLIBC_2.0) [1]	pututxline(GLIBC_2.1) [1]	utmpname(GLIBC_2.0) [2]
getgrgid(GLIBC_2.0) [1]	getprotobynumber(GLIBC_2.0) [1]	getservbyport(GLIBC_2.0) [1]	setgrent(GLIBC_2.0) [1]	

191

192

Referenced Specification(s)

193

[1]. ISO POSIX (2003)

194

[2]. this specification

195

[3]. SUSv2

endgrent(GLIBC_2.0) [SUSv3]	endprotoent(GLIBC_2.0) [SUSv3]	endpwent(GLIBC_2.0) [SUSv3]	endservent(GLIBC_2.0) [SUSv3]
endutent(GLIBC_2.0) [SUSv2]	endutxent(GLIBC_2.1) [SUSv3]	getgrent(GLIBC_2.0) [SUSv3]	getgrgid(GLIBC_2.0) [SUSv3]
getgrgid_r(GLIBC_2.1.2) [SUSv3]	getgrnam(GLIBC_2.0) [SUSv3]	getgrnam_r(GLIBC_2.1.2) [SUSv3]	getgrouplist(GLIBC_2.2.4) [LSB]
gethostbyaddr(GLIBC_2.0) [SUSv3]	gethostbyname(GLIBC_2.0) [SUSv3]	getprotobyname(GLIBC_2.0) [SUSv3]	getprotobynumber(GLIBC_2.0) [SUSv3]
getprotoent(GLIBC_2.0) [SUSv3]	getpwent(GLIBC_2.0) [SUSv3]	getpwnam(GLIBC_2.0) [SUSv3]	getpwnam_r(GLIBC_2.1.2) [SUSv3]
getpwuid(GLIBC_2.0) [SUSv3]	getpwuid_r(GLIBC_2.1.2) [SUSv3]	getservbyname(GLIBC_2.0) [SUSv3]	getservbyport(GLIBC_2.0) [SUSv3]
getservent(GLIBC_2.0) [SUSv3]	getutent(GLIBC_2.0) [LSB]	getutent_r(GLIBC_2.0) [LSB]	getutxent(GLIBC_2.1) [SUSv3]
getutxid(GLIBC_2.1) [SUSv3]	getutxline(GLIBC_2.1) [SUSv3]	pututxline(GLIBC_2.1) [SUSv3]	setgrent(GLIBC_2.0) [SUSv3]
setgroups(GLIBC_2.0) [LSB]	setprotoent(GLIBC_2.0) [SUSv3]	setpwent(GLIBC_2.0) [SUSv3]	setservent(GLIBC_2.0) [SUSv3]
setutent(GLIBC_2.0) [LSB]	setutxent(GLIBC_2.1) [SUSv3]	utmpname(GLIBC_2.0) [LSB]	

196

11.2.15 Language Support

197

11.2.15.1 Interfaces for Language Support

198

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.

199

200

201

Table 11-20 libc - Language Support Function Interfaces

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

202

203

Referenced Specification(s)

204

[1], this specification

205

__libc_start_main(GLIBC_2.0) [LSB]			
-----------------------------------------------	--	--	--

11.2.16 Large File Support

206

11.2.16.1 Interfaces for Large File Support

207

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.

208

209

210

Table 11-21 libc - Large File Support Function Interfaces

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

211

Referenced Specification(s)

212

[1], this specification

213

[2], Large File Support

214

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

215

11.2.17 Standard Library

216

11.2.17.1 Interfaces for Standard Library

217

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.

218

219

220

Table 11-22 libc - Standard Library Function Interfaces

<code>_Exit(GLIBC_2.1.1)</code> [1]	<code>dirname(GLIBC_2.0)</code> [1]	<code>gettimeofday(GLIBC_2.0)</code> [1]	<code>lrand48(GLIBC_2.0)</code> [1]	<code>srand(GLIBC_2.0)</code> [1]
<code>__assert_fail(GLIBC_2.0)</code> [2]	<code>div(GLIBC_2.0)</code> [1]	<code>glob(GLIBC_2.0)</code> [1]	<code>lsearch(GLIBC_2.0)</code> [1]	<code>srand48(GLIBC_2.0)</code> [1]
<code>__exa_atexit()</code>	<code>drand48(GLIBC_2.0)</code> [1]	<code>glob64(GLIBC_2.0)</code> [1]	<code>makecontext(GLIBC_2.0)</code> [1]	<code>srandom(GLIBC_2.0)</code> [1]

GLIBC_2.1.3) [2]	BC_2.0)[1]	_2.1)[2]	GLIBC_2.1) [1]	BC_2.0)[1]
__errno_location(GLIBC_2.0))[2]	ecvt(GLIBC_2.0)[1]	globfree(GLIBC_2.0)[1]	malloc(GLIBC_2.0)[1]	strtod(GLIBC_2.0)[1]
__fpending(GLIBC_2.2)[2]	erand48(GLIBC_2.0)[1]	globfree64(GLIBC_2.1)[2]	memmem(GLIBC_2.0)[2]	strtol(GLIBC_2.0)[1]
__getpagesize(GLIBC_2.0))[2]	err(GLIBC_2.0)[2]	grantpt(GLIBC_2.1)[1]	mkstemp(GLIBC_2.0)[1]	strtoul(GLIBC_2.0)[1]
__isinf(GLIBC_2.0)[2]	error(GLIBC_2.0)[2]	herecreate(GLIBC_2.0)[1]	mktemp(GLIBC_2.0)[1]	swapcontext(GLIBC_2.1) [1]
__isinf(GLIBC_2.0)[2]	errx(GLIBC_2.0)[2]	hdestroy(GLIBC_2.0)[1]	mrand48(GLIBC_2.0)[1]	syslog(GLIBC_2.0)[1]
__isnfl(GLIBC_2.0)[2]	fevt(GLIBC_2.0)[1]	hsearch(GLIBC_2.0)[1]	nftw(GLIBC_2.3.3)[1]	system(GLIBC_2.0)[2]
__isnan(GLIBC_2.0)[2]	fmtmsg(GLIBC_2.1)[1]	htonl(GLIBC_2.0)[1]	nrand48(GLIBC_2.0)[1]	tdelete(GLIBC_2.0)[1]
__isnanf(GLIBC_2.0)[2]	fnmatch(GLIBC_2.2.3)[1]	htons(GLIBC_2.0)[1]	ntohl(GLIBC_2.0)[1]	tfind(GLIBC_2.0)[1]
__isnanl(GLIBC_2.0)[2]	fpathconf(GLIBC_2.0)[1]	imaxabs(GLIBC_2.1.1)[1]	ntohs(GLIBC_2.0)[1]	tmpfile(GLIBC_2.1)[1]
__sysconf(GLIBC_2.2)[2]	free(GLIBC_2.0)[1]	imaxdiv(GLIBC_2.1.1)[1]	openlog(GLIBC_2.0)[1]	tmpnam(GLIBC_2.0)[1]
_exit(GLIBC_2.0)[1]	freeaddrinfo(GLIBC_2.0) [1]	inet_addr(GLIBC_2.0)[1]	perror(GLIBC_2.0)[1]	tsearch(GLIBC_2.0)[1]
_longjmp(GLIBC_2.0)[1]	ftrylockfile(GLIBC_2.0)[1]	inet_ntoa(GLIBC_2.0)[1]	posix_memalign(GLIBC_2.2))[1]	ttyname(GLIBC_2.0)[1]
__setjmp(GLIBC_2.0)[1]	ftw(GLIBC_2.0)[1]	inet_ntop(GLIBC_2.0)[1]	posix_openpt(GLIBC_2.2.1) [1]	ttyname_r(GLIBC_2.0)[1]
a64l(GLIBC_2.0)[1]	funlockfile(GLIBC_2.0)[1]	inet_pton(GLIBC_2.0)[1]	ptsname(GLIBC_2.1)[1]	twalk(GLIBC_2.0)[1]
abort(GLIBC_2.0)[1]	gai_strerror(GLIBC_2.1)[1]	initstate(GLIBC_2.0)[1]	putenv(GLIBC_2.0)[1]	unlockpt(GLIBC_2.1)[1]
abs(GLIBC_2.0)[1]	gevt(GLIBC_2.0)[1]	insque(GLIBC_2.0)[1]	qsort(GLIBC_2.0)[1]	unsetenv(GLIBC_2.0)[1]
atof(GLIBC_2.0)[1]	getaddrinfo(GLIBC_2.0)[1]	isatty(GLIBC_2.0)[1]	rand(GLIBC_2.0)[1]	usleep(GLIBC_2.0)[1]

atoi(GLIBC_2.0) [1]	getcwd(GLIBC_2.0) [1]	isblank(GLIBC_2.0) [1]	rand_r(GLIBC_2.0) [1]	verrx(GLIBC_2.0) [2]
atol(GLIBC_2.0) [1]	getdate(GLIBC_2.1) [1]	jrand48(GLIBC_2.0) [1]	random(GLIBC_2.0) [1]	vfscanf(GLIBC_2.0) [2]
atoll(GLIBC_2.0) [1]	getenv(GLIBC_2.0) [1]	l64a(GLIBC_2.0) [1]	realloc(GLIBC_2.0) [1]	vscanf(GLIBC_2.0) [2]
basename(GLIBC_2.0) [1]	getlogin(GLIBC_2.0) [1]	labs(GLIBC_2.0) [1]	realpath(GLIBC_2.3) [1]	vsscanf(GLIBC_2.0) [2]
bsearch(GLIBC_2.0) [1]	getlogin_r(GLIBC_2.0) [1]	lcong48(GLIBC_2.0) [1]	remque(GLIBC_2.0) [1]	vsyslog(GLIBC_2.0) [2]
calloc(GLIBC_2.0) [1]	getnameinfo(GLIBC_2.1) [1]	ldiv(GLIBC_2.0) [1]	seed48(GLIBC_2.0) [1]	warn(GLIBC_2.0) [2]
closelog(GLIBC_2.0) [1]	getopt(GLIBC_2.0) [2]	lfind(GLIBC_2.0) [1]	setenv(GLIBC_2.0) [1]	warnx(GLIBC_2.0) [2]
confstr(GLIBC_2.0) [1]	getopt_long(GLIBC_2.0) [2]	llabs(GLIBC_2.0) [1]	sethostname(GLIBC_2.0) [2]	wordexp(GLIBC_2.1) [1]
cuserid(GLIBC_2.0) [3]	getopt_long_only(GLIBC_2.0) [2]	lldiv(GLIBC_2.0) [1]	setlogmask(GLIBC_2.0) [1]	wordfree(GLIBC_2.1) [1]
daemon(GLIBC_2.0) [2]	getsubopt(GLIBC_2.0) [1]	longjmp(GLIBC_2.0) [1]	setstate(GLIBC_2.0) [1]	

221

Referenced Specification(s)

222

[1]

223

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

drand48(GLIBC_2.0) [SUSv3]	ecvt(GLIBC_2.0) [SUSv3]	erand48(GLIBC_2.0) [SUSv3]	err(GLIBC_2.0) [LSB]
error(GLIBC_2.0) [LSB]	errx(GLIBC_2.0) [LSB]	fcvt(GLIBC_2.0) [SUSv3]	fmtmsg(GLIBC_2.1) [SUSv3]
fnmatch(GLIBC_2.3) [SUSv3]	fpathconf(GLIBC_2.0) [SUSv3]	free(GLIBC_2.0) [SUSv3]	freaddrinfo(GLIBC_2.0) [SUSv3]
ftrylockfile(GLIBC_2.0) [SUSv3]	ftw(GLIBC_2.0) [SUSv3]	funlockfile(GLIBC_2.0) [SUSv3]	gai_strerror(GLIBC_2.1) [SUSv3]
gcvt(GLIBC_2.0) [SUSv3]	getaddrinfo(GLIBC_2.0) [SUSv3]	getcwd(GLIBC_2.0) [SUSv3]	getdate(GLIBC_2.1) [SUSv3]
getenv(GLIBC_2.0) [SUSv3]	getlogin(GLIBC_2.0) [SUSv3]	getlogin_r(GLIBC_2.0) [SUSv3]	getnameinfo(GLIBC_2.1) [SUSv3]
getopt(GLIBC_2.0) [LSB]	getopt_long(GLIBC_2.0) [LSB]	getopt_long_only(GLIBC_2.0) [LSB]	getsubopt(GLIBC_2.0) [SUSv3]
gettimeofday(GLIBC_2.0) [SUSv3]	glob(GLIBC_2.0) [SUSv3]	glob64(GLIBC_2.1) [LSB]	globfree(GLIBC_2.0) [SUSv3]
globfree64(GLIBC_2.1) [LSB]	grantpt(GLIBC_2.1) [SUSv3]	hcreate(GLIBC_2.0) [SUSv3]	hdestroy(GLIBC_2.0) [SUSv3]
hsearch(GLIBC_2.0) [SUSv3]	htonl(GLIBC_2.0) [SUSv3]	htons(GLIBC_2.0) [SUSv3]	imaxabs(GLIBC_2.1.1) [SUSv3]
imaxdiv(GLIBC_2.1.1) [SUSv3]	inet_addr(GLIBC_2.0) [SUSv3]	inet_ntoa(GLIBC_2.0) [SUSv3]	inet_ntop(GLIBC_2.0) [SUSv3]
inet_pton(GLIBC_2.0) [SUSv3]	initstate(GLIBC_2.0) [SUSv3]	insque(GLIBC_2.0) [SUSv3]	isatty(GLIBC_2.0) [SUSv3]
isblank(GLIBC_2.0) [SUSv3]	jrand48(GLIBC_2.0) [SUSv3]	l64a(GLIBC_2.0) [SUSv3]	labs(GLIBC_2.0) [SUSv3]
lcong48(GLIBC_2.0) [SUSv3]	ldiv(GLIBC_2.0) [SUSv3]	lfind(GLIBC_2.0) [SUSv3]	llabs(GLIBC_2.0) [SUSv3]
lldiv(GLIBC_2.0) [SUSv3]	longjmp(GLIBC_2.0) [SUSv3]	lrand48(GLIBC_2.0) [SUSv3]	lsearch(GLIBC_2.0) [SUSv3]
makecontext(GLIBC_2.1) [SUSv3]	malloc(GLIBC_2.0) [SUSv3]	memmem(GLIBC_2.0) [LSB]	mkstemp(GLIBC_2.0) [SUSv3]
mktemp(GLIBC_2.0) [SUSv3]	mrnd48(GLIBC_2.0) [SUSv3]	nftw(GLIBC_2.3.3) [SUSv3]	nrnd48(GLIBC_2.0) [SUSv3]
ntohl(GLIBC_2.0) [SUSv3]	ntohs(GLIBC_2.0) [SUSv3]	openlog(GLIBC_2.0) [SUSv3]	perror(GLIBC_2.0) [SUSv3]
posix_memalign(GLIBC_2.2) [SUSv3]	posix_openpt(GLIBC_2.2.1) [SUSv3]	ptsname(GLIBC_2.1) [SUSv3]	putenv(GLIBC_2.0) [SUSv3]
qsort(GLIBC_2.0) [SUSv3]	rand(GLIBC_2.0) [SUSv3]	rand_r(GLIBC_2.0) [SUSv3]	random(GLIBC_2.0) [SUSv3]

realloc(GLIBC_2.0) [SUSv3]	realpath(GLIBC_2.3) [SUSv3]	remque(GLIBC_2.0) [SUSv3]	seed48(GLIBC_2.0) [SUSv3]
setenv(GLIBC_2.0) [SUSv3]	sethostname(GLIBC_2.0) [LSB]	setlogmask(GLIBC_2.0) [SUSv3]	setstate(GLIBC_2.0) [SUSv3]
srand(GLIBC_2.0) [SUSv3]	srand48(GLIBC_2.0) [SUSv3]	srandom(GLIBC_2.0) [SUSv3]	strtod(GLIBC_2.0) [SUSv3]
strtol(GLIBC_2.0) [SUSv3]	strtoul(GLIBC_2.0) [SUSv3]	swapcontext(GLIBC_2.1) [SUSv3]	syslog(GLIBC_2.0) [SUSv3]
system(GLIBC_2.0) [LSB]	tdelete(GLIBC_2.0) [SUSv3]	tfind(GLIBC_2.0) [SUSv3]	tmpfile(GLIBC_2.1) [SUSv3]
tmpnam(GLIBC_2.0) [SUSv3]	tsearch(GLIBC_2.0) [SUSv3]	ttynam(GLIBC_2.0) [SUSv3]	ttynam_r(GLIBC_2.0) [SUSv3]
twalk(GLIBC_2.0) [SUSv3]	unlockpt(GLIBC_2.1) [SUSv3]	unsetenv(GLIBC_2.0) [SUSv3]	usleep(GLIBC_2.0) [SUSv3]
verrx(GLIBC_2.0) [LSB]	vfscanf(GLIBC_2.0) [LSB]	vscanf(GLIBC_2.0) [LSB]	vsscanf(GLIBC_2.0) [LSB]
vsyslog(GLIBC_2.0) [LSB]	warn(GLIBC_2.0) [LSB]	warnx(GLIBC_2.0) [LSB]	wordexp(GLIBC_2.1) [SUSv3]
wordfree(GLIBC_2.1) [SUSv3]			

224

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

227 ~~[2]. this specification~~

228 ~~[3]. SUSv2~~

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

232 **Table 11-23 libc - Standard Library Data Interfaces**

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

233

234 *Referenced Specification(s)*

235 ~~[1]. this specification~~

236 ~~[2]. ISO POSIX (2003)~~

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

optopt(GLIBC_2.0)[SUSv3]			
------------------------------	--	--	--

237

11.3 Data Definitions for libc

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

242 ~~These definitions are intended to supplement those provided in~~ Where an interface
 243 is defined as requiring a particular system header file all of the ~~referenced~~
 244 ~~underlying~~ data definitions for that system header file presented here shall be in
 245 effect.

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

250 This specification uses ~~ISO/IEC 9899~~ the ISO C (1999) C Language as the reference
 251 programming language, and data definitions are specified in ISO C format. The C
 252 language is used here as a convenient notation. Using a C language description of
 253 these data objects does not preclude their use by other programming languages.

11.3.1 arpa/inet.h

```
254
255 extern uint32_t htonl(uint32_t);
256 extern uint16_t htons(uint16_t);
257 extern in_addr_t inet_addr(const char *);
258 extern char *inet_ntoa(struct in_addr);
259 extern const char *inet_ntop(int, const void *, char *, socklen_t);
260 extern int inet_pton(int, const char *, void *);
261 extern uint32_t ntohl(uint32_t);
262 extern uint16_t ntohs(uint16_t);
```

11.3.2 assert.h

```
263
264 extern void __assert_fail(const char *, const char *, unsigned int,
265                          const char *);
```

11.3.3 ctype.h

```
266
267 extern int _tolower(int);
268 extern int _toupper(int);
269 extern int isalnum(int);
270 extern int isalpha(int);
271 extern int isascii(int);
272 extern int iscntrl(int);
273 extern int isdigit(int);
274 extern int isgraph(int);
275 extern int islower(int);
276 extern int isprint(int);
277 extern int ispunct(int);
278 extern int isspace(int);
279 extern int isupper(int);
280 extern int isxdigit(int);
```

```

281     extern int toascii(int);
282     extern int tolower(int);
283     extern int toupper(int);
284     extern int isblank(int);
285     extern const unsigned short **__ctype_b_loc(void);
286     extern const int32_t **__ctype_toupper_loc(void);
287     extern const int32_t **__ctype_tolower_loc(void);

```

11.3.4 dirent.h

```

288
289     extern void rewinddir(DIR *);
290     extern void seekdir(DIR *, long int);
291     extern long int telldir(DIR *);
292     extern int closedir(DIR *);
293     extern DIR *opendir(const char *);
294     extern struct dirent *readdir(DIR *);
295     extern struct dirent64 *readdir64(DIR *);
296     extern int readdir_r(DIR *, struct dirent *, struct dirent **);

```

11.3.5 err.h

```

297
298     extern void err(int, const char *, ...);
299     extern void errx(int, const char *, ...);
300     extern void warn(const char *, ...);
301     extern void warnx(const char *, ...);
302     extern void error(int, int, const char *, ...);

```

11.3.6 errno.h

```

303
304     #define EDEADLOCK      EDEADLK
305
306     extern int *__errno_location(void);

```

11.3.27 fcntl.h

```

307
308     #define F_GETLK64      12
309     #define F_SETLK64      13
310     #define F_SETLKW64    14
311
312     extern int lockf64(int, int, off64_t);
313     extern int fcntl(int, int, ...);

```

11.3.8 fmtmsg.h

```

314
315     extern int fmtmsg(long int, const char *, int, const char *, const char
316     *,
317                     const char *);

```

11.3.9 fnmatch.h

```

318
319     extern int fnmatch(const char *, const char *, int);

```

11.3.10 ftw.h

```

320
321     extern int ftw(const char *, __ftw_func_t, int);

```

```

322 extern int ftw64(const char *, __ftw64_func_t, int);
323 extern int nftw(const char *, __nftw_func_t, int, int);
324 extern int nftw64(const char *, __nftw64_func_t, int, int);

```

11.3.11 getopt.h

```

325
326 extern int getopt_long(int, char *const, const char *,
327                       const struct option *, int *);
328 extern int getopt_long_only(int, char *const, const char *,
329                             const struct option *, int *);

```

11.3.12 glob.h

```

330
331 extern int glob(const char *, int,
332               int (*__errfunc) (const char *p1, int p2)
333               , glob_t *);
334 extern int glob64(const char *, int,
335                 int (*__errfunc) (const char *p1, int p2)
336                 , glob64_t *);
337 extern void globfree(glob_t *);
338 extern void globfree64(glob64_t *);

```

11.3.13 grp.h

```

339
340 extern void endgrent(void);
341 extern struct group *getgrent(void);
342 extern struct group *getgrgid(gid_t);
343 extern struct group *getgrnam(char *);
344 extern int initgroups(const char *, gid_t);
345 extern void setgrent(void);
346 extern int setgroups(size_t, const gid_t *);
347 extern int getgrgid_r(gid_t, struct group *, char *, size_t,
348                     struct group **);
349 extern int getgrnam_r(const char *, struct group *, char *, size_t,
350                     struct group **);
351 extern int getgrouplist(const char *, gid_t, gid_t *, int *);

```

11.3.14 iconv.h

```

352
353 extern size_t iconv(iconv_t, char **, size_t *, char **, size_t *);
354 extern int iconv_close(iconv_t);
355 extern iconv_t iconv_open(char *, char *);

```

11.3.15 inttypes.h

```

356
357 typedef unsigned long long int uint64_t;
358 typedef long long int intmax_t;
359 typedef unsigned long long int uintmax_t;
360 typedef unsigned int uintptr_t;
361
362 extern intmax_t strtoumax(const char *, char **, int);
363 extern uintmax_t strtoumax(const char *, char **, int);
364 extern intmax_t wcstoumax(const wchar_t *, wchar_t * *, int);
365 extern uintmax_t wcstoumax(const wchar_t *, wchar_t * *, int);
366 extern intmax_t imaxabs(intmax_t);
367 extern imaxdiv_t imaxdiv(intmax_t, intmax_t);

```

11.3.416 langinfo.h

```
368 extern char *nl_langinfo(nl_item);
369
```

11.3.17 libgen.h

```
370
371 extern char *basename(const char *);
372 extern char *dirname(char *);
```

11.3.18 libintl.h

```
373
374 extern char *bindtextdomain(const char *, const char *);
375 extern char *dcgettext(const char *, const char *, int);
376 extern char *dgettext(const char *, const char *);
377 extern char *gettext(const char *);
378 extern char *textdomain(const char *);
379 extern char *bind_textdomain_codeset(const char *, const char *);
380 extern char *dcngettext(const char *, const char *, const char *,
381                        unsigned long int, int);
382 extern char *dngettext(const char *, const char *, const char *,
383                        unsigned long int);
384 extern char *ngettext(const char *, const char *, unsigned long int);
```

11.3.19 limits.h

```
385
386 #define ULONG_MAX      0xFFFFFFFFUL
387 #define LONG_MAX       2147483647
388
389 #define CHAR_MIN      0
390 #define CHAR_MAX      255
391
392 #define PTHREAD_STACK_MIN      16384
```

11.3.5 setjmp.h

```
393
394 typedef int __jmp_buf[14];
```

11.3.6 signal.h

```
395
396 #define __NUM_ACRS      16
397 #define __NUM_FPRS      16
398 #define __NUM_GPRS      16
399
400 typedef struct
401 {
402     unsigned long int mask;
403     unsigned long int addr;
404 } __attribute__((aligned(8))) __psw_t;
405 typedef struct
406 {
407     __psw_t psw;
408     unsigned long int gprs[__NUM_GPRS];
409     unsigned int acrs[__NUM_ACRS];
410 }
411 __s390_regs_common;
```

```

413 #define SICEV_PAD_SIZE ((SICEV_MAX_SIZE/sizeof(int))-3)
414
415 #define SI_PAD_SIZE ((SI_MAX_SIZE/sizeof(int))-3)
416
417 struct sigaction
418 {
419     union
420     {
421         sighandler_t __sa_handler;
422         void (*__sa_sigaction) (int, siginfo_t *, void *);
423     }
424     __sigaction_handler;
425     sigset_t sa_mask;
426     unsigned long int sa_flags;
427     void (*sa_restorer) (void);
428 }
429
430 #define MINSIGSTKSZ 2048
431 #define SIGSTKSZ 8192
432
433 typedef struct
434 {
435     unsigned int fpe;
436     double fprs[__NUM_FPRS];
437 }
438 __s390_fp_regs;
439 typedef struct
440 {
441     __s390_regs_common regs;
442     __s390_fp_regs fpregs;
443 }
444 __sigregs;
445
446 struct sigcontext
447 {
448     unsigned long int oldmask[2];
449     __sigregs *sregs;
450 }
451

```

11.3.7 stddef.h

```

452
453 typedef unsigned long int size_t;
454 typedef int ptrdiff_t;

```

11.3.8 stdio.h

```

455
456 #define __IO_FILE_SIZE 152

```

11.3.9 sys/ioctl.h

```

457
458 #define TIOCGWINSZ 0x5413
459 #define FIONREAD 0x541B
460 #define TIOCNOTTY 21538

```

11.3.10 sys/ipc.h

```

461
462 struct ipc_perm
463 {

```

```

464     __key_t __key;
465     __uid_t uid;
466     __gid_t gid;
467     __uid_t cuid;
468     __uid_t egid;
469     __unsigned short mode;
470     __unsigned short __pad1;
471     __unsigned short __seq;
472     __unsigned short __pad2;
473     __unsigned long int __unused1;
474     __unsigned long int __unused2;
475     };
476     };

```

11.3.11 sys/mman.h

```

477
478     #define MCL_CURRENT 1
479     #define MCL_FUTURE 2

```

11.3.12 sys/msg.h

```

480
481     typedef unsigned long int msglen_t;
482     typedef unsigned long int msgqnum_t;
483
484     struct msgid_ds
485     {
486         __struct ipc_perm msg_perm;
487         __time_t msg_stime;
488         __unsigned long int __unused1;
489         __time_t msg_rtime;
490         __unsigned long int __unused2;
491         __time_t msg_etime;
492         __unsigned long int __unused3;
493         __unsigned long int __msg_cbytes;
494         __msgqnum_t msg_qnum;
495         __msglen_t msg_qbytes;
496         __pid_t msg_lspid;
497         __pid_t msg_lrpid;
498         __unsigned long int __unused4;
499         __unsigned long int __unused5;
500     };
501     };

```

11.3.13 sys/sem.h

11.3.20 locale.h

```

502
503     extern struct lconv *localeconv(void);
504     extern char *setlocale(int, const char *);
505     extern locale_t uselocale(locale_t);
506     extern void freelocale(locale_t);
507     extern locale_t duplocale(locale_t);
508     extern locale_t newlocale(int, const char *, locale_t);

```

11.3.21 monetary.h

```

509
510     extern ssize_t strfmon(char *, size_t, const char *, ...);

```


11.3.22 net/if.h

```

511
512 struct semid_ds
513 {
514     struct ipc_perm sem_perm;
515     time_t sem_otime;
516     unsigned long int __unused1;
517     time_t sem_ctime;
518     unsigned long int __unused2;
519     unsigned long int sem_nsems;
520     unsigned long int __unused3;
521     unsigned long int __unused4;
522 }
523 ;

```

11.3.14 sys/shm.h

```

524
525 #define SHMLBA (__getpagesize())
526
527 typedef unsigned long int shmatt_t;
528
529 struct shmid_ds
530 {
531     struct ipc_perm shm_perm;
532     size_t shm_segsz;
533     time_t shm_atime;
534     unsigned long int __unused1;
535     time_t shm_dtime;
536     unsigned long int __unused2;
537     time_t shm_ctime;
538     unsigned long int __unused3;
539     pid_t shm_epid;
540     pid_t shm_lpid;
541     shmatt_t shm_nattch;
542     unsigned long int __unused4;
543     unsigned long int __unused5;
544 }
545 ;

```

11.3.15 sys/socket.h

```

546
547 typedef uint32_t __cs_aligntype;
548
549 #define SO_RCVLOWAT 18
550 #define SO_SNDLOWAT 19
551 #define SO_RCVTIMEO 20
552 #define SO_SNDTIMEO 21

```

11.3.16 sys/stat.h

```

553
554 #define __STAT_VER 3
555
556 struct stat
557 {
558     dev_t st_dev;
559     unsigned int __pad1;
560     ino_t st_ino;
561     mode_t st_mode;
562     nlink_t st_nlink;

```

```

563     __uid_t st_uid;
564     __gid_t st_gid;
565     __dev_t st_rdev;
566     __unsigned int __pad2;
567     __off_t st_size;
568     __blksize_t st_blksize;
569     __blkcnt_t st_blocks;
570     struct timespec st_atim;
571     struct timespec st_mtim;
572     struct timespec st_ctim;
573     __unsigned long int __unused4;
574     __unsigned long int __unused5;
575 }
576 ↵
577 struct stat64
578 {
579     __dev_t st_dev;
580     __int __pad1;
581     __ino_t st_ino;
582     __mode_t st_mode;
583     __nlink_t st_nlink;
584     __uid_t st_uid;
585     __gid_t st_gid;
586     __dev_t st_rdev;
587     __int __pad2;
588     __off64_t st_size;
589     __blksize_t st_blksize;
590     __blkcnt64_t st_blocks;
591     struct timespec st_atim;
592     struct timespec st_mtim;
593     struct timespec st_ctim;
594     __ino64_t st_ino;
595 }
596 ↵

```

11.3.17 sys/statvfs.h

```

597 extern void if_freenameindex(struct if_nameindex *);
598 extern char *if_indextoname(unsigned int, char *);
599 extern struct if_nameindex *if_nameindex(void);
600 extern unsigned int if_nametoindex(const char *);

```

11.3.23 netdb.h

```

601
602 extern void endprotoent(void);
603 extern void endservent(void);
604 extern void freeaddrinfo(struct addrinfo *);
605 extern const char *gai_strerror(int);
606 extern int getaddrinfo(const char *, const char *, const struct addrinfo
607 *,
608                       struct addrinfo **);
609 extern struct hostent *gethostbyaddr(const void *, socklen_t, int);
610 extern struct hostent *gethostbyname(const char *);
611 extern struct protoent *getprotobyname(const char *);
612 extern struct protoent *getprotobynumber(int);
613 extern struct protoent *getprotoent(void);
614 extern struct servent *getservbyname(const char *, const char *);
615 extern struct servent *getservbyport(int, const char *);
616 extern struct servent *getservent(void);
617 extern void setprotoent(int);
618 extern void setservent(int);
619 extern int *__h_errno_location(void);

```

11.3.24 netinet/in.h

```
620 extern int bindresvport(int, struct sockaddr_in *);
621
```

11.3.25 netinet/ip.h

```
622
623 /*
624  * This header is architecture neutral
625  * Please refer to the generic specification for details
626  */
```

11.3.26 netinet/tcp.h

```
627
628 /*
629  * This header is architecture neutral
630  * Please refer to the generic specification for details
631  */
```

11.3.27 netinet/udp.h

```
632
633 /*
634  * This header is architecture neutral
635  * Please refer to the generic specification for details
636  */
```

11.3.28 nl_types.h

```
637
638 extern int catclose(nl_catd);
639 extern char *catgets(nl_catd, int, int, const char *);
640 extern nl_catd catopen(const char *, int);
```

11.3.29 poll.h

```
641
642 extern int poll(struct pollfd *, nfd_t, int);
```

11.3.30 pty.h

```
643
644 struct statvfs
645 {
646     unsigned long int f_bsize;
647     unsigned long int f_frsize;
648     fsblkent_t f_blocks;
649     fsblkent_t f_bfree;
650     fsblkent_t f_bavail;
651     fsfilent_t f_files;
652     fsfilent_t f_ffree;
653     fsfilent_t f_favail;
654     unsigned long int f_fsid;
655     int __f_unused;
656     unsigned long int f_flag;
657     unsigned long int f_namemax;
658     int __f_spare[6];
659 }
660 ;
661 struct statvfs64
```

```

662     {
663     — unsigned long int f_bsize;
664     — unsigned long int f_frsize;
665     — fsblkcnt64_t f_blocks;
666     — fsblkcnt64_t f_bfree;
667     — fsblkcnt64_t f_bavail;
668     — fsfilcnt64_t f_files;
669     — fsfilcnt64_t f_ffree;
670     — fsfilcnt64_t f_favail;
671     — unsigned long int f_fsid;
672     — int __f_unused;
673     — unsigned long int f_flag;
674     — unsigned long int f_namemax;
675     — int __f_spare[6];
676     }
677     ;

```

11.3.18 sys/types.h

```

678
679     typedef long long int int64_t;
680
681     typedef int32_t ssize_t;
682
683     #define __FDSET_LONGS 32

```

11.3.19 termios.h

```

684
685     #define OLCUC 0000002
686     #define ONLCR 0000004
687     #define XCASE 0000004
688     #define NLDLY 0000400
689     #define CR1 0001000
690     #define IUCLC 0001000
691     #define CR2 0002000
692     #define CR3 0003000
693     #define CRDLY 0003000
694     #define TAB1 0004000
695     #define TAB2 0010000
696     #define TAB3 0014000
697     #define TABDLY 0014000
698     #define BS1 0020000
699     #define BSDLY 0020000
700     #define VT1 0040000
701     #define VTDLY 0040000
702     #define FF1 0100000
703     #define FFDLY 0100000
704
705     #define VSUSP 10
706     #define VEOL 11
707     #define VREPRINT 12
708     #define VDISCARD 13
709     #define VWERASE 14
710     #define VEOL2 16
711     #define VMIN 6
712     #define VSWTC 7
713     #define VSTART 8
714     #define VSTOP 9
715
716     #define IXON 0002000
717     #define IXOFF 0010000
718
719     #define CS6 0000020

```

```

720 #define CS7 0000040
721 #define CS8 0000060
722 #define CSIZE 0000060
723 #define CSTOPB 0000100
724 #define CREAD 0000200
725 #define PARENB 0000400
726 #define PARODD 0001000
727 #define HUPCL 0002000
728 #define CLOCAL 0004000
729 #define VTIME 5
730
731 #define ISIG 0000001
732 #define ICANON 0000002
733 #define ECHOE 0000020
734 #define ECHOK 0000040
735 #define ECHONL 0000100
736 #define NOFLSH 0000200
737 #define TOSTOP 0000400
738 #define ECHOCTL 0001000
739 #define ECHOPRT 0002000
740 #define ECHOKE 0004000
741 #define FLUSHO 0010000
742 #define PENDIN 0040000
743 #define IEXTEN 0100000

```

11.3.20 ucontext.h

```

744
745 #define NREG 36
746
747 typedef union
748 {
749     double d;
750     float f;
751 }
752 fpreg_t;
753 extern int openpty(int *, int *, char *, struct termios *,
754                  struct winsize *);
755 extern int forkpty(int *, char *, struct termios *, struct winsize *);

```

11.3.31 pwd.h

```

756
757 extern void endpwent(void);
758 extern struct passwd *getpwent(void);
759 extern struct passwd *getpwnam(char *);
760 extern struct passwd *getpwuid(uid_t);
761 extern void setpwent(void);
762 extern int getpwnam_r(char *, struct passwd *, char *, size_t,
763                    struct passwd **);
764 extern int getpwuid_r(uid_t, struct passwd *, char *, size_t,
765                    struct passwd **);

```

11.3.32 regex.h

```

766
767 extern int regcomp(regex_t *, const char *, int);
768 extern size_t regerror(int, const regex_t *, char *, size_t);
769 extern int regexexec(const regex_t *, const char *, size_t, regmatch_t,
770                    int);
771 extern void regfree(regex_t *);

```

11.3.33 rpc/auth.h

```

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

```

11.3.34 rpc/clnt.h

```

776 extern struct CLIENT *clnt_create(const char *, const u_long, const
777 u_long,
778                                     const char *);
779 extern void clnt_pcreateerror(const char *);
780 extern void clnt_perrno(enum clnt_stat);
781 extern void clnt_perror(struct CLIENT *, const char *);
782 extern char *clnt_spcreateerror(const char *);
783 extern char *clnt_sperrno(enum clnt_stat);
784 extern char *clnt_spperror(struct CLIENT *, const char *);
785

```

11.3.35 rpc/pmap_clnt.h

```

786 extern u_short pmap_getport(struct sockaddr_in *, const u_long,
787                             const u_long, u_int);
788 extern bool_t pmap_set(const u_long, const u_long, int, u_short);
789 extern bool_t pmap_unset(u_long, u_long);
790

```

11.3.36 rpc/rpc_msg.h

```

791 extern bool_t xdr_callhdr(XDR *, struct rpc_msg *);
792

```

11.3.37 rpc/svc.h

```

793 extern void svc_getreqset(fd_set *);
794 extern bool_t svc_register(SVCXPRT *, rpcprog_t, rpcvers_t,
795                             __dispatch_fn_t, rpcprot_t);
796 extern void svc_run(void);
797 extern bool_t svc_sendreply(SVCXPRT *, xdrproc_t, caddr_t);
798 extern void svcerr_auth(SVCXPRT *, enum auth_stat);
799 extern void svcerr_decode(SVCXPRT *);
800 extern void svcerr_noproc(SVCXPRT *);
801 extern void svcerr_noprogram(SVCXPRT *);
802 extern void svcerr_progvers(SVCXPRT *, rpcvers_t, rpcvers_t);
803 extern void svcerr_systemerr(SVCXPRT *);
804 extern void svcerr_weakauth(SVCXPRT *);
805 extern SVCXPRT *svctcp_create(int, u_int, u_int);
806 extern SVCXPRT *svcudp_create(int);
807

```

11.3.38 rpc/types.h

```

808 /*
809  * This header is architecture neutral
810  * Please refer to the generic specification for details
811  */
812

```

11.3.39 rpc/xdr.h

```

813

```

```

814 extern bool_t xdr_array(XDR *, caddr_t *, u_int *, u_int, u_int,
815                       xdrproc_t);
816 extern bool_t xdr_bool(XDR *, bool_t *);
817 extern bool_t xdr_bytes(XDR *, char **, u_int *, u_int);
818 extern bool_t xdr_char(XDR *, char *);
819 extern bool_t xdr_double(XDR *, double *);
820 extern bool_t xdr_enum(XDR *, enum_t *);
821 extern bool_t xdr_float(XDR *, float *);
822 extern void xdr_free(xdrproc_t, char *);
823 extern bool_t xdr_int(XDR *, int *);
824 extern bool_t xdr_long(XDR *, long int *);
825 extern bool_t xdr_opaque(XDR *, caddr_t, u_int);
826 extern bool_t xdr_pointer(XDR *, char **, u_int, xdrproc_t);
827 extern bool_t xdr_reference(XDR *, caddr_t *, u_int, xdrproc_t);
828 extern bool_t xdr_short(XDR *, short *);
829 extern bool_t xdr_string(XDR *, char **, u_int);
830 extern bool_t xdr_u_char(XDR *, u_char *);
831 extern bool_t xdr_u_int(XDR *, u_int *);
832 extern bool_t xdr_u_long(XDR *, u_long *);
833 extern bool_t xdr_u_short(XDR *, u_short *);
834 extern bool_t xdr_union(XDR *, enum_t *, char *,
835                       const struct xdr_discrim *, xdrproc_t);
836 extern bool_t xdr_vector(XDR *, char *, u_int, u_int, xdrproc_t);
837 extern bool_t xdr_void(void);
838 extern bool_t xdr_wrapstring(XDR *, char **);
839 extern void xdrmem_create(XDR *, caddr_t, u_int, enum xdr_op);
840 extern void xdrrec_create(XDR *, u_int, u_int, caddr_t,
841                          int (*__readit) (char *p1, char *p2, int p3)
842                          , int (*__writeit) (char *p1, char *p2, int
843                          p3)
844                          );
845 extern typedef int bool_t xdrrec_eof(XDR *);

```

11.3.40 sched.h

```

846 extern int sched_get_priority_max(int);
847 extern int sched_get_priority_min(int);
848 extern int sched_getparam(pid_t, struct sched_param *);
849 extern int sched_getscheduler(pid_t);
850 extern int sched_rr_get_interval(pid_t, struct timespec *);
851 extern int sched_setparam(pid_t, const struct sched_param *);
852 extern int sched_setscheduler(pid_t, int, const struct sched_param *);
853 extern int sched_yield(void);
854

```

11.3.41 search.h

```

855
856 typedef struct
857 {
858     unsigned int fpe;
859     fpreg_t fprs[16];
860 }
861 fpregset_t;
862
863 typedef struct
864 {
865     _psw_t psw;
866     unsigned long int gregs[16];
867     unsigned int aregs[16];
868     fpregset_t fpregs;
869 }
870 mecontext_t;
871 extern int hcreate(size_t);

```

```

872 extern ENTRY *hsearch(ENTRY, ACTION);
873 extern void insque(void *, void *);
874 extern void *lfind(const void *, const void *, size_t *, size_t,
875                  __compar_fn_t);
876 extern void *lsearch(const void *, void *, size_t *, size_t,
877                    __compar_fn_t);
878 extern void remque(void *);
879 extern void hdestroy(void);
880 extern void *tdelete(const void *, void **, __compar_fn_t);
881 extern void *tfind(const void *, void *const *, __compar_fn_t);
882 extern void *tsearch(const void *, void **, __compar_fn_t);
883 extern void twalk(const void *, __action_fn_t);

```

11.3.42 setjmp.h

```

884
885 typedef int __jmp_buf[14];
886
887 typedef struct ucontext
888 {
889     unsigned long int uc_flags;
890     struct ucontext *uc_link;
891     stack_t uc_stack;
892     mccontext_t uc_mccontext;
893     sigset_t uc_sigmask;
894 }
895 ucontext_t;

```

11.3.21 unistd.h

```

896 extern int __sigsetjmp(jmp_buf, int);
897 extern void longjmp(jmp_buf, int);
898 extern void siglongjmp(sigjmp_buf, int);
899 extern void _longjmp(jmp_buf, int);
900 extern int _setjmp(jmp_buf);

```

11.3.43 signal.h

```

901
902 #define __NUM_ACRS      16
903 #define __NUM_FPRS     16
904 #define __NUM_GPRS     16
905
906 typedef struct {
907     unsigned long int mask;
908     unsigned long int addr;
909 } __attribute__((aligned(8)))
910   _psw_t;
911 typedef struct {
912     _psw_t psw;
913     unsigned long int gprs[__NUM_GPRS];
914     unsigned int acrs[__NUM_ACRS];
915 } _s390_regs_common;
916
917 #define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int))-3)
918
919 #define SI_PAD_SIZE     ((SI_MAX_SIZE/sizeof(int))-3)
920
921 struct sigaction {
922     union {
923         sighandler_t _sa_handler;
924         void (*_sa_sigaction)(int, siginfo_t *, void *);
925     } __sigaction_handler;
926     sigset_t sa_mask;

```



```

927         unsigned long int sa_flags;
928         void (*sa_restorer) (void);
929     };
930
931     #define MINSIGSTKSZ      2048
932     #define SIGSTKSZ        8192
933
934     typedef struct {
935         unsigned int fpc;
936         double fprs[__NUM_FPRS];
937     } _s390_fp_regs;
938     typedef struct {
939         _s390_regs_common regs;
940         _s390_fp_regs fpregs;
941     } _sigregs;
942
943     struct sigcontext {
944         unsigned long int oldmask[2];
945         _sigregs *sregs;
946     };
947     extern int __libc_current_sigrtmax(void);
948     extern int __libc_current_sigrtmin(void);
949     extern sighandler_t __sysv_signal(int, sighandler_t);
950     extern char *const _sys_siglist(void);
951     extern int killpg(pid_t, int);
952     extern void psignal(int, const char *);
953     extern int raise(int);
954     extern int sigaddset(sigset_t *, int);
955     extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
956     extern int sigdelset(sigset_t *, int);
957     extern int sigemptyset(sigset_t *);
958     extern int sigfillset(sigset_t *);
959     extern int sighold(int);
960     extern int sigignore(int);
961     extern int siginterrupt(int, int);
962     extern int sigisemptyset(const sigset_t *);
963     extern int sigismember(const sigset_t *, int);
964     extern int sigorset(sigset_t *, const sigset_t *, const sigset_t *);
965     extern int sigpending(sigset_t *);
966     extern int sigrelse(int);
967     extern sighandler_t sigset(int, sighandler_t);
968     extern int pthread_kill(pthread_t, int);
969     extern int pthread_sigmask(int, sigset_t *, sigset_t *);
970     extern int sigaction(int, const struct sigaction *, struct sigaction *);
971     extern int sigwait(sigset_t *, int *);
972     extern int kill(pid_t, int);
973     extern int sigaltstack(const struct sigaltstack *, struct sigaltstack
974     *);
975     extern sighandler_t signal(int, sighandler_t);
976     extern int sigpause(int);
977     extern int sigprocmask(int, const sigset_t *, sigset_t *);
978     extern int sigreturn(struct sigcontext *);
979     extern int sigsuspend(const sigset_t *);
980     extern int sigqueue(pid_t, int, const union sigval);
981     extern int sigwaitinfo(const sigset_t *, siginfo_t *);
982     extern int sigtimedwait(const sigset_t *, siginfo_t *,
983         const struct timespec *);
984     extern sighandler_t bsd_signal(int, sighandler_t);

```

11.3.44 stddef.h

```

985
986     typedef unsigned long int size_t;
987     typedef int ptrdiff_t;

```

11.3.45 stdio.h

```

988
989     #define __IO_FILE_SIZE 152
990
991     extern char *const _sys_errlist(void);
992     extern void clearerr(FILE *);
993     extern int fclose(FILE *);
994     extern FILE *fdopen(int, const char *);
995     extern int fflush_unlocked(FILE *);
996     extern int fileno(FILE *);
997     extern FILE *fopen(const char *, const char *);
998     extern int fprintf(FILE *, const char *, ...);
999     extern int fputc(int, FILE *);
1000     extern FILE *freopen(const char *, const char *, FILE *);
1001     extern FILE *freopen64(const char *, const char *, FILE *);
1002     extern int fscanf(FILE *, const char *, ...);
1003     extern int fseek(FILE *, long int, int);
1004     extern int fseeko(FILE *, off_t, int);
1005     extern int fseeko64(FILE *, loff_t, int);
1006     extern off_t ftello(FILE *);
1007     extern loff_t ftello64(FILE *);
1008     extern int getchar(void);
1009     extern int getchar_unlocked(void);
1010     extern int getw(FILE *);
1011     extern int pclose(FILE *);
1012     extern void perror(const char *);
1013     extern FILE *popen(const char *, const char *);
1014     extern int printf(const char *, ...);
1015     extern int putc_unlocked(int, FILE *);
1016     extern int putchar(int);
1017     extern int putchar_unlocked(int);
1018     extern int putw(int, FILE *);
1019     extern int remove(const char *);
1020     extern void rewind(FILE *);
1021     extern int scanf(const char *, ...);
1022     extern void setbuf(FILE *, char *);
1023     extern int sprintf(char *, const char *, ...);
1024     extern int sscanf(const char *, const char *, ...);
1025     extern FILE *stderr(void);
1026     extern FILE *stdin(void);
1027     extern FILE *stdout(void);
1028     extern char *tempnam(const char *, const char *);
1029     extern FILE *tmpfile64(void);
1030     extern FILE *tmpfile(void);
1031     extern char *tmpnam(char *);
1032     extern int vfprintf(FILE *, const char *, va_list);
1033     extern int vprintf(const char *, va_list);
1034     extern int feof(FILE *);
1035     extern int ferror(FILE *);
1036     extern int fflush(FILE *);
1037     extern int fgetc(FILE *);
1038     extern int fgetpos(FILE *, fpos_t *);
1039     extern char *fgets(char *, int, FILE *);
1040     extern int fputs(const char *, FILE *);
1041     extern size_t fread(void *, size_t, size_t, FILE *);
1042     extern int fsetpos(FILE *, const fpos_t *);
1043     extern long int ftell(FILE *);
1044     extern size_t fwrite(const void *, size_t, size_t, FILE *);
1045     extern int getc(FILE *);
1046     extern int putc(int, FILE *);
1047     extern int puts(const char *);
1048     extern int setvbuf(FILE *, char *, int, size_t);
1049     extern int snprintf(char *, size_t, const char *, ...);

```

```

1050 extern int ungetc(int, FILE *);
1051 extern int vsnprintf(char *, size_t, const char *, va_list);
1052 extern int vsprintf(char *, const char *, va_list);
1053 extern void flockfile(FILE *);
1054 extern int asprintf(char **, const char *, ...);
1055 extern int fgetpos64(FILE *, fpos64_t *);
1056 extern FILE *fopen64(const char *, const char *);
1057 extern int fsetpos64(FILE *, const fpos64_t *);
1058 extern int ftrylockfile(FILE *);
1059 extern void funlockfile(FILE *);
1060 extern int getc_unlocked(FILE *);
1061 extern void setbuffer(FILE *, char *, size_t);
1062 extern int vasprintf(char **, const char *, va_list);
1063 extern int vdprintf(int, const char *, va_list);
1064 extern int vfscanf(FILE *, const char *, va_list);
1065 extern int vscanf(const char *, va_list);
1066 extern int vsscanf(const char *, const char *, va_list);
1067 extern size_t __fpending(FILE *);

```

11.3.46 stdlib.h

```

1068
1069 extern double __strtod_internal(const char *, char **, int);
1070 extern float __strtof_internal(const char *, char **, int);
1071 extern long int __strtoul_internal(const char *, char **, int, int);
1072 extern long double __strtold_internal(const char *, char **, int);
1073 extern long long int __strtoll_internal(const char *, char **, int, int);
1074 extern unsigned long int __strtoul_internal(const char *, char **, int,
1075                                             int);
1076 extern unsigned long long int __strtoull_internal(const char *, char **,
1077                                                  int, int);
1078 extern long int a64l(const char *);
1079 extern void abort(void);
1080 extern int abs(int);
1081 extern double atof(const char *);
1082 extern int atoi(char *);
1083 extern long int atol(char *);
1084 extern long long int atoll(const char *);
1085 extern void *bsearch(const void *, const void *, size_t, size_t,
1086                     __compar_fn_t);
1087 extern div_t div(int, int);
1088 extern double drand48(void);
1089 extern char *ecvt(double, int, int *, int *);
1090 extern double erand48(unsigned short);
1091 extern void exit(int);
1092 extern char *fcvt(double, int, int *, int *);
1093 extern char *gcvt(double, int, char *);
1094 extern char *getenv(const char *);
1095 extern int getsuopt(char **, char *const *, char **);
1096 extern int grantpt(int);
1097 extern long int jrand48(unsigned short);
1098 extern char *l64a(long int);
1099 extern long int labs(long int);
1100 extern void lcong48(unsigned short);
1101 extern ldiv_t ldiv(long int, long int);
1102 extern long long int llabs(long long int);
1103 extern lldiv_t lldiv(long long int, long long int);
1104 extern long int lrand48(void);
1105 extern int mblen(const char *, size_t);
1106 extern size_t mbstowcs(wchar_t *, const char *, size_t);
1107 extern int mbtowc(wchar_t *, const char *, size_t);
1108 extern char *mktemp(char *);
1109 extern long int mrand48(void);
1110 extern long int nrand48(unsigned short);

```

```

1111 extern char *ptsname(int);
1112 extern int putenv(char *);
1113 extern void qsort(void *, size_t, size_t, __compar_fn_t);
1114 extern int rand(void);
1115 extern int rand_r(unsigned int *);
1116 extern unsigned short *seed48(unsigned short);
1117 extern void srand48(long int);
1118 extern int unlockpt(int);
1119 extern size_t wcstombs(char *, const wchar_t *, size_t);
1120 extern int wctomb(char *, wchar_t);
1121 extern int system(const char *);
1122 extern void *calloc(size_t, size_t);
1123 extern void free(void *);
1124 extern char *initstate(unsigned int, char *, size_t);
1125 extern void *malloc(size_t);
1126 extern long int random(void);
1127 extern void *realloc(void *, size_t);
1128 extern char *setstate(char *);
1129 extern void srand(unsigned int);
1130 extern void srand48(unsigned int);
1131 extern double strtod(char *, char **);
1132 extern float strtof(const char *, char **);
1133 extern long int strtol(char *, char **, int);
1134 extern long double strtold(const char *, char **);
1135 extern long long int strtoll(const char *, char **, int);
1136 extern long long int strtoll(const char *, char **, int);
1137 extern unsigned long int strtoul(const char *, char **, int);
1138 extern unsigned long long int strtoull(const char *, char **, int);
1139 extern unsigned long long int strtouq(const char *, char **, int);
1140 extern void _Exit(int);
1141 extern size_t __ctype_get_mb_cur_max(void);
1142 extern char **environ(void);
1143 extern char *realpath(const char *, char *);
1144 extern int setenv(const char *, const char *, int);
1145 extern int unsetenv(const char *);
1146 extern int getloadavg(double, int);
1147 extern int mkstemp64(char *);
1148 extern int posix_memalign(void **, size_t, size_t);
1149 extern int posix_openpt(int);

```

11.3.47 string.h

```

1150
1151 extern void *__memcpy(void *, const void *, size_t);
1152 extern char *__strcpy(char *, const char *);
1153 extern char *__strtok_r(char *, const char *, char **);
1154 extern void bcopy(void *, void *, size_t);
1155 extern void *memchr(void *, int, size_t);
1156 extern int memcmp(void *, void *, size_t);
1157 extern void *memcpy(void *, void *, size_t);
1158 extern void *memmem(const void *, size_t, const void *, size_t);
1159 extern void *memmove(void *, const void *, size_t);
1160 extern void *memset(void *, int, size_t);
1161 extern char *strcat(char *, const char *);
1162 extern char *strchr(char *, int);
1163 extern int strcmp(char *, char *);
1164 extern int strcoll(const char *, const char *);
1165 extern char *strcpy(char *, char *);
1166 extern size_t strcspn(const char *, const char *);
1167 extern char *strerror(int);
1168 extern size_t strlen(char *);
1169 extern char *strncat(char *, char *, size_t);
1170 extern int strncmp(char *, char *, size_t);
1171 extern char *strncpy(char *, char *, size_t);

```

```

1172 extern char *strpbrk(const char *, const char *);
1173 extern char *strrchr(char *, int);
1174 extern char *strsignal(int);
1175 extern size_t strspn(const char *, const char *);
1176 extern char *strstr(char *, char *);
1177 extern char *strtok(char *, const char *);
1178 extern size_t strxfrm(char *, const char *, size_t);
1179 extern int bcmp(void *, void *, size_t);
1180 extern void bzero(void *, size_t);
1181 extern int ffs(int);
1182 extern char *index(char *, int);
1183 extern void *memccpy(void *, const void *, int, size_t);
1184 extern char *rindex(char *, int);
1185 extern int strcasecmp(char *, char *);
1186 extern char *strdup(char *);
1187 extern int strncasecmp(char *, char *, size_t);
1188 extern char *strndup(const char *, size_t);
1189 extern size_t strnlen(const char *, size_t);
1190 extern char *strsep(char **, const char *);
1191 extern char *strerror_r(int, char *, size_t);
1192 extern char *strtok_r(char *, const char *, char **);
1193 extern char *strcasestr(const char *, const char *);
1194 extern char *stpcpy(char *, const char *);
1195 extern char *stpncpy(char *, const char *, size_t);
1196 extern void *memrchr(const void *, int, size_t);

```

11.3.48 sys/file.h

```

1197
1198 extern int flock(int, int);

```

11.3.49 sys/ioctl.h

```

1199
1200 #define TIOCGWINSZ      0x5413
1201 #define FIONREAD        0x541B
1202 #define TIOCNOTTY      21538
1203
1204 extern int ioctl(int, unsigned long int, ...);

```

11.3.50 sys/ipc.h

```

1205
1206 struct ipc_perm {
1207     key_t __key;
1208     uid_t uid;
1209     gid_t gid;
1210     uid_t cuid;
1211     uid_t cgid;
1212     unsigned short mode;
1213     unsigned short __pad1;
1214     unsigned short __seq;
1215     unsigned short __pad2;
1216     unsigned long int __unused1;
1217     unsigned long int __unused2;
1218 };
1219
1220 extern key_t ftok(char *, int);

```

11.3.51 sys/mman.h

```

1221
1222 #define MCL_CURRENT      1

```

```

1223     #define MCL_FUTURE      2
1224
1225     extern int msync(void *, size_t, int);
1226     extern int mlock(const void *, size_t);
1227     extern int mlockall(int);
1228     extern void *mmap(void *, size_t, int, int, int, off_t);
1229     extern int mprotect(void *, size_t, int);
1230     extern int munlock(const void *, size_t);
1231     extern int munlockall(void);
1232     extern int munmap(void *, size_t);
1233     extern void *mmap64(void *, size_t, int, int, int, off64_t);
1234     extern int shm_open(const char *, int, mode_t);
1235     extern int shm_unlink(const char *);

```

11.3.52 sys/msg.h

```

1236
1237     typedef unsigned long int msglen_t;
1238     typedef unsigned long int msgqnum_t;
1239
1240     struct msqid_ds {
1241         struct ipc_perm msg_perm;
1242         time_t msg_stime;
1243         unsigned long int __unused1;
1244         time_t msg_rtime;
1245         unsigned long int __unused2;
1246         time_t msg_ctime;
1247         unsigned long int __unused3;
1248         unsigned long int __msg_cbytes;
1249         msgqnum_t msg_qnum;
1250         msglen_t msg_qbytes;
1251         pid_t msg_lspid;
1252         pid_t msg_lrpid;
1253         unsigned long int __unused4;
1254         unsigned long int __unused5;
1255     };
1256     extern int msgctl(int, int, struct msqid_ds *);
1257     extern int msgget(key_t, int);
1258     extern int msgrcv(int, void *, size_t, long int, int);
1259     extern int msgsnd(int, const void *, size_t, int);

```

11.3.53 sys/param.h

```

1260
1261     /*
1262     * This header is architecture neutral
1263     * Please refer to the generic specification for details
1264     */

```

11.3.54 sys/poll.h

```

1265
1266     /*
1267     * This header is architecture neutral
1268     * Please refer to the generic specification for details
1269     */

```

11.3.55 sys/resource.h

```

1270
1271     extern int getpriority(__priority_which_t, id_t);
1272     extern int getrlimit64(id_t, struct rlimit64 *);
1273     extern int setpriority(__priority_which_t, id_t, int);

```

```

1274 extern int setrlimit(__rlimit_resource_t, const struct rlimit *);
1275 extern int setrlimit64(__rlimit_resource_t, const struct rlimit64 *);
1276 extern int getrlimit(__rlimit_resource_t, struct rlimit *);
1277 extern int getrusage(int, struct rusage *);

```

11.3.56 sys/sem.h

```

1278
1279 struct semid_ds {
1280     struct ipc_perm sem_perm;
1281     time_t sem_otime;
1282     unsigned long int __unused1;
1283     time_t sem_ctime;
1284     unsigned long int __unused2;
1285     unsigned long int sem_nsems;
1286     unsigned long int __unused3;
1287     unsigned long int __unused4;
1288 };
1289 extern int semctl(int, int, int, ...);
1290 extern int semget(key_t, int, int);
1291 extern int semop(int, struct sembuf *, size_t);

```

11.3.57 sys/shm.h

```

1292
1293 #define SHMLBA (__getpagesize())
1294
1295 typedef unsigned long int shmatt_t;
1296
1297 struct shmid_ds {
1298     struct ipc_perm shm_perm;
1299     size_t shm_segsz;
1300     time_t shm_atime;
1301     unsigned long int __unused1;
1302     time_t shm_dtime;
1303     unsigned long int __unused2;
1304     time_t shm_ctime;
1305     unsigned long int __unused3;
1306     pid_t shm_cpid;
1307     pid_t shm_lpid;
1308     shmatt_t shm_nattch;
1309     unsigned long int __unused4;
1310     unsigned long int __unused5;
1311 };
1312 extern int __getpagesize(void);
1313 extern void *shmat(int, const void *, int);
1314 extern int shmctl(int, int, struct shmid_ds *);
1315 extern int shmdt(const void *);
1316 extern int shmget(key_t, size_t, int);

```

11.3.58 sys/socket.h

```

1317
1318 typedef uint32_t __ss_aligntype;
1319
1320 #define SO_RCVLOWAT 18
1321 #define SO_SNDLOWAT 19
1322 #define SO_RCVTIMEO 20
1323 #define SO_SNDTIMEO 21
1324
1325 extern int bind(int, const struct sockaddr *, socklen_t);
1326 extern int getnameinfo(const struct sockaddr *, socklen_t, char *,
1327                        socklen_t, char *, socklen_t, unsigned int);

```

```

1328     extern int getsockname(int, struct sockaddr *, socklen_t *);
1329     extern int listen(int, int);
1330     extern int setsockopt(int, int, int, const void *, socklen_t);
1331     extern int accept(int, struct sockaddr *, socklen_t *);
1332     extern int connect(int, const struct sockaddr *, socklen_t);
1333     extern ssize_t recv(int, void *, size_t, int);
1334     extern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *,
1335                             socklen_t *);
1336     extern ssize_t recvmsg(int, struct msghdr *, int);
1337     extern ssize_t send(int, const void *, size_t, int);
1338     extern ssize_t sendmsg(int, const struct msghdr *, int);
1339     extern ssize_t sendto(int, const void *, size_t, int,
1340                           const struct sockaddr *, socklen_t);
1341     extern int getpeername(int, struct sockaddr *, socklen_t *);
1342     extern int getsockopt(int, int, int, void *, socklen_t *);
1343     extern int shutdown(int, int);
1344     extern int socket(int, int, int);
1345     extern int socketpair(int, int, int, int);
1346     extern int sockatmark(int);

```

11.3.59 sys/stat.h

```

1347
1348     #define _STAT_VER          3
1349
1350     struct stat {
1351         dev_t st_dev;
1352         unsigned int __pad1;
1353         ino_t st_ino;
1354         mode_t st_mode;
1355         nlink_t st_nlink;
1356         uid_t st_uid;
1357         gid_t st_gid;
1358         dev_t st_rdev;
1359         unsigned int __pad2;
1360         off_t st_size;
1361         blksize_t st_blksize;
1362         blkcnt_t st_blocks;
1363         struct timespec st_atim;
1364         struct timespec st_mtim;
1365         struct timespec st_ctim;
1366         unsigned long int __unused4;
1367         unsigned long int __unused5;
1368     };
1369     struct stat64 {
1370         dev_t st_dev;
1371         int __pad1;
1372         ino_t __st_ino;
1373         mode_t st_mode;
1374         nlink_t st_nlink;
1375         uid_t st_uid;
1376         gid_t st_gid;
1377         dev_t st_rdev;
1378         int __pad2;
1379         off64_t st_size;
1380         blksize_t st_blksize;
1381         blkcnt64_t st_blocks;
1382         struct timespec st_atim;
1383         struct timespec st_mtim;
1384         struct timespec st_ctim;
1385         ino64_t st_ino;
1386     };
1387
1388     extern int __fxstat(int, int, struct stat *);

```



```

1389 extern int __fxstat64(int, int, struct stat64 *);
1390 extern int __lxstat(int, char *, struct stat *);
1391 extern int __lxstat64(int, const char *, struct stat64 *);
1392 extern int __xmknod(int, const char *, mode_t, dev_t *);
1393 extern int __xstat(int, const char *, struct stat *);
1394 extern int __xstat64(int, const char *, struct stat64 *);
1395 extern int mkfifo(const char *, mode_t);
1396 extern int chmod(const char *, mode_t);
1397 extern int fchmod(int, mode_t);
1398 extern mode_t umask(mode_t);

```

11.3.60 sys/statvfs.h

```

1399
1400 struct statvfs {
1401     unsigned long int f_bsize;
1402     unsigned long int f_frsize;
1403     fsblkcnt_t f_blocks;
1404     fsblkcnt_t f_bfree;
1405     fsblkcnt_t f_bavail;
1406     fsfilcnt_t f_files;
1407     fsfilcnt_t f_ffree;
1408     fsfilcnt_t f_favail;
1409     unsigned long int f_fsid;
1410     int __f_unused;
1411     unsigned long int f_flag;
1412     unsigned long int f_namemax;
1413     int __f_spare[6];
1414 };
1415 struct statvfs64 {
1416     unsigned long int f_bsize;
1417     unsigned long int f_frsize;
1418     fsblkcnt64_t f_blocks;
1419     fsblkcnt64_t f_bfree;
1420     fsblkcnt64_t f_bavail;
1421     fsfilcnt64_t f_files;
1422     fsfilcnt64_t f_ffree;
1423     fsfilcnt64_t f_favail;
1424     unsigned long int f_fsid;
1425     int __f_unused;
1426     unsigned long int f_flag;
1427     unsigned long int f_namemax;
1428     int __f_spare[6];
1429 };
1430 extern int fstatvfs(int, struct statvfs *);
1431 extern int fstatvfs64(int, struct statvfs64 *);
1432 extern int statvfs(const char *, struct statvfs *);
1433 extern int statvfs64(const char *, struct statvfs64 *);

```

11.3.61 sys/time.h

```

1434
1435 extern int getitimer(__itimer_which_t, struct itimerval *);
1436 extern int setitimer(__itimer_which_t, const struct itimerval *,
1437                     struct itimerval *);
1438 extern int adjtime(const struct timeval *, struct timeval *);
1439 extern int gettimeofday(struct timeval *, struct timezone *);
1440 extern int utimes(const char *, const struct timeval *);

```

11.3.62 sys/timex.h

```

1441
1442 extern int ftime(struct timeb *);

```

11.3.63 sys/times.h

```
1443 extern clock_t times(struct tms *);
1444
```

11.3.64 sys/types.h

```
1445
1446 typedef long long int int64_t;
1447
1448 typedef int32_t ssize_t;
1449
1450 #define __FDSET_LONGS 32
```

11.3.65 sys/uio.h

```
1451
1452 extern ssize_t readv(int, const struct iovec *, int);
1453 extern ssize_t writev(int, const struct iovec *, int);
```

11.3.66 sys/un.h

```
1454
1455 /*
1456  * This header is architecture neutral
1457  * Please refer to the generic specification for details
1458  */
```

11.3.67 sys/utsname.h

```
1459
1460 extern int uname(struct utsname *);
```

11.3.68 sys/wait.h

```
1461
1462 extern pid_t wait(int *);
1463 extern pid_t waitpid(pid_t, int *, int);
1464 extern pid_t wait4(pid_t, int *, int, struct rusage *);
```

11.3.69 syslog.h

```
1465
1466 extern void closelog(void);
1467 extern void openlog(const char *, int, int);
1468 extern int setlogmask(int);
1469 extern void syslog(int, const char *, ...);
1470 extern void vsyslog(int, const char *, va_list);
```

11.3.70 termios.h

```
1471
1472 #define OLCUC 0000002
1473 #define ONLCR 0000004
1474 #define XCASE 0000004
1475 #define NLDLY 0000400
1476 #define CR1 0001000
1477 #define IUCLC 0001000
1478 #define CR2 0002000
1479 #define CR3 0003000
1480 #define CRDLY 0003000
1481 #define TAB1 0004000
```

```

1482     #define TAB2      0010000
1483     #define TAB3      0014000
1484     #define TABDLY    0014000
1485     #define BS1       0020000
1486     #define BSDLY     0020000
1487     #define VT1       0040000
1488     #define VTDLY     0040000
1489     #define FF1       0100000
1490     #define FFDLY     0100000
1491
1492     #define VSUSP     10
1493     #define VEOL      11
1494     #define VREPRINT      12
1495     #define VDISCARD     13
1496     #define VWERASE     14
1497     #define VEOL2      16
1498     #define VMIN       6
1499     #define VSWTC      7
1500     #define VSTART     8
1501     #define VSTOP      9
1502
1503     #define IXON       0002000
1504     #define IXOFF      0010000
1505
1506     #define CS6        0000020
1507     #define CS7        0000040
1508     #define CS8        0000060
1509     #define CSIZE      0000060
1510     #define CSTOPB    0000100
1511     #define CREAD      0000200
1512     #define PARENB     0000400
1513     #define PARODD     0001000
1514     #define HUPCL      0002000
1515     #define CLOCAL     0004000
1516     #define VTIME      5
1517
1518     #define ISIG       0000001
1519     #define ICANON     0000002
1520     #define ECHOE      0000020
1521     #define ECHOK      0000040
1522     #define ECHONL     0000100
1523     #define NOFLSH     0000200
1524     #define TOSTOP     0000400
1525     #define ECHOCTL    0001000
1526     #define ECHOPRT    0002000
1527     #define ECHOKE     0004000
1528     #define FLUSHO     0010000
1529     #define PENDIN     0040000
1530     #define IEXTEN     0100000
1531
1532     extern speed_t cfgetispeed(const struct termios *);
1533     extern speed_t cfgetospeed(const struct termios *);
1534     extern void cfmakeraw(struct termios *);
1535     extern int cfsetispeed(struct termios *, speed_t);
1536     extern int cfsetospeed(struct termios *, speed_t);
1537     extern int cfsetspeed(struct termios *, speed_t);
1538     extern int tcflow(int, int);
1539     extern int tcflush(int, int);
1540     extern pid_t tcgetsid(int);
1541     extern int tcsendbreak(int, int);
1542     extern int tcsetattr(int, int, const struct termios *);
1543     extern int tcdrain(int);
1544     extern int tcgetattr(int, struct termios *);

```

11.3.71 time.h

```

1545 extern int __daylight(void);
1546 extern long int __timezone(void);
1547 extern char *__tzname(void);
1548 extern char *asctime(const struct tm *);
1549 extern clock_t clock(void);
1550 extern char *ctime(const time_t *);
1551 extern char *ctime_r(const time_t *, char *);
1552 extern double difftime(time_t, time_t);
1553 extern struct tm *getdate(const char *);
1554 extern int getdate_err(void);
1555 extern struct tm *gmtime(const time_t *);
1556 extern struct tm *localtime(const time_t *);
1557 extern time_t mktime(struct tm *);
1558 extern int stime(const time_t *);
1559 extern size_t strftime(char *, size_t, const char *, const struct tm *);
1560 extern char *strptime(const char *, const char *, struct tm *);
1561 extern time_t time(time_t *);
1562 extern int nanosleep(const struct timespec *, struct timespec *);
1563 extern int daylight(void);
1564 extern long int timezone(void);
1565 extern char *tzname(void);
1566 extern void tzset(void);
1567 extern char *asctime_r(const struct tm *, char *);
1568 extern struct tm *gmtime_r(const time_t *, struct tm *);
1569 extern struct tm *localtime_r(const time_t *, struct tm *);
1570 extern int clock_getcpuclockid(pid_t, clockid_t *);
1571 extern int clock_getres(clockid_t, struct timespec *);
1572 extern int clock_gettime(clockid_t, struct timespec *);
1573 extern int clock_nanosleep(clockid_t, int, const struct timespec *,
1574                             struct timespec *);
1575 extern int clock_settime(clockid_t, const struct timespec *);
1576 extern int timer_create(clockid_t, struct sigevent *, timer_t *);
1577 extern int timer_delete(timer_t);
1578 extern int timer_getoverrun(timer_t);
1579 extern int timer_gettime(timer_t, struct itimerspec *);
1580 extern int timer_settime(timer_t, int, const struct itimerspec *,
1581                             struct itimerspec *);

```

11.3.72 ucontext.h

```

1583
1584 #define NGREG 36
1585
1586 typedef union {
1587     double d;
1588     float f;
1589 } fpreg_t;
1590
1591 typedef struct {
1592     unsigned int fpc;
1593     fpreg_t fprs[16];
1594 } fpregset_t;
1595
1596 typedef struct {
1597     _psw_t psw;
1598     unsigned long int gregs[16];
1599     unsigned int aregs[16];
1600     fpregset_t fpregs;
1601 } mcontext_t;
1602
1603 typedef struct ucontext {

```

```

1604         unsigned long int uc_flags;
1605         struct ucontext *uc_link;
1606         stack_t uc_stack;
1607         mcontext_t uc_mcontext;
1608         sigset_t uc_sigmask;
1609     } ucontext_t;
1610 extern int getcontext(ucontext_t *);
1611 extern int makecontext(ucontext_t *, void (*func) (void)
1612                      , int, ...);
1613 extern int setcontext(const struct ucontext *);
1614 extern int swapcontext(ucontext_t *, const struct ucontext *);

```

11.3.73 ulimit.h

```

1615
1616 extern long int ulimit(int, ...);

```

11.3.74 unistd.h

```

1617
1618 typedef int intptr_t;
1619
1620 extern char **__environ(void);
1621 extern pid_t __getpgid(pid_t);
1622 extern void _exit(int);
1623 extern int acct(const char *);
1624 extern unsigned int alarm(unsigned int);
1625 extern int chown(const char *, uid_t, gid_t);
1626 extern int chroot(const char *);
1627 extern size_t confstr(int, char *, size_t);
1628 extern int creat(const char *, mode_t);
1629 extern int creat64(const char *, mode_t);
1630 extern char *ctermid(char *);
1631 extern char *cuserid(char *);
1632 extern int daemon(int, int);
1633 extern int execl(const char *, const char *, ...);
1634 extern int execlp(const char *, const char *, ...);
1635 extern int execlp(const char *, const char *, ...);
1636 extern int execvp(const char *, char *const);
1637 extern int execvp(const char *, char *const);
1638 extern int fdatsync(int);
1639 extern int ftruncate64(int, off64_t);
1640 extern long int gethostid(void);
1641 extern char *getlogin(void);
1642 extern int getlogin_r(char *, size_t);
1643 extern int getopt(int, char *const, const char *);
1644 extern pid_t getpgrp(void);
1645 extern pid_t getsid(pid_t);
1646 extern char *getwd(char *);
1647 extern int lockf(int, int, off_t);
1648 extern int mkstemp(char *);
1649 extern int nice(int);
1650 extern char *optarg(void);
1651 extern int opterr(void);
1652 extern int optind(void);
1653 extern int optopt(void);
1654 extern int rename(const char *, const char *);
1655 extern int setegid(gid_t);
1656 extern int seteuid(uid_t);
1657 extern int sethostname(const char *, size_t);
1658 extern int setpgrp(void);
1659 extern void swab(const void *, void *, ssize_t);
1660 extern void sync(void);
1661 extern pid_t tcgetpgrp(int);

```

```

1662     extern int tcsetpgrp(int, pid_t);
1663     extern int truncate(const char *, off_t);
1664     extern int truncate64(const char *, off64_t);
1665     extern char *ttyname(int);
1666     extern unsigned int ualarm(useconds_t, useconds_t);
1667     extern int usleep(useconds_t);
1668     extern int close(int);
1669     extern int fsync(int);
1670     extern off_t lseek(int, off_t, int);
1671     extern int open(const char *, int, ...);
1672     extern int pause(void);
1673     extern ssize_t read(int, void *, size_t);
1674     extern ssize_t write(int, const void *, size_t);
1675     extern char *crypt(char *, char *);
1676     extern void encrypt(char *, int);
1677     extern void setkey(const char *);
1678     extern int access(const char *, int);
1679     extern int brk(void *);
1680     extern int chdir(const char *);
1681     extern int dup(int);
1682     extern int dup2(int, int);
1683     extern int execve(const char *, char *const, char *const);
1684     extern int fchdir(int);
1685     extern int fchown(int, uid_t, gid_t);
1686     extern pid_t fork(void);
1687     extern gid_t getegid(void);
1688     extern uid_t geteuid(void);
1689     extern gid_t getgid(void);
1690     extern int getgroups(int, gid_t);
1691     extern int gethostname(char *, size_t);
1692     extern pid_t getpgid(pid_t);
1693     extern pid_t getpid(void);
1694     extern uid_t getuid(void);
1695     extern int lchown(const char *, uid_t, gid_t);
1696     extern int link(const char *, const char *);
1697     extern int mkdir(const char *, mode_t);
1698     extern long int pathconf(const char *, int);
1699     extern int pipe(int);
1700     extern int readlink(const char *, char *, size_t);
1701     extern int rmdir(const char *);
1702     extern void *sbrk(ptrdiff_t);
1703     extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
1704     extern int setgid(gid_t);
1705     extern int setpgid(pid_t, pid_t);
1706     extern int setregid(gid_t, gid_t);
1707     extern int setreuid(uid_t, uid_t);
1708     extern pid_t setsid(void);
1709     extern int setuid(uid_t);
1710     extern unsigned int sleep(unsigned int);
1711     extern int symlink(const char *, const char *);
1712     extern long int sysconf(int);
1713     extern int unlink(const char *);
1714     extern pid_t vfork(void);
1715     extern ssize_t pread(int, void *, size_t, off_t);
1716     extern ssize_t pwrite(int, const void *, size_t, off_t);
1717     extern char **_environ(void);
1718     extern long int fpathconf(int, int);
1719     extern int ftruncate(int, off_t);
1720     extern char *getcwd(char *, size_t);
1721     extern int getpagesize(void);
1722     extern pid_t getppid(void);
1723     extern int isatty(int);
1724     extern loff_t lseek64(int, loff_t, int);
1725     extern int open64(const char *, int, ...);

```

```

1726 extern ssize_t pread64(int, void *, size_t, off64_t);
1727 extern ssize_t pwrite64(int, const void *, size_t, off64_t);
1728 extern int ttyname_r(int, char *, size_t);

```

11.3.2275 utime.h

```

1729
1730 extern int utime(const char *, const struct utimbuf *);

```

11.3.76 utmp.h

```

1731
1732 struct lastlog {
1733     †
1734     time_t ll_time;
1735     char ll_line[UT_LINESIZE];
1736     char ll_host[UT_HOSTSIZE];
1737 }
1738 -;
1739
1740 struct utmp {
1741     †
1742     short ut_type;
1743     pid_t ut_pid;
1744     char ut_line[UT_LINESIZE];
1745     char ut_id[4];
1746     char ut_user[UT_NAMESIZE];
1747     char ut_host[UT_HOSTSIZE];
1748     struct exit_status ut_exit;
1749     long int ut_session;
1750     struct timeval ut_tv;
1751     int32_t ut_addr_v6[4];
1752     char __unused[20];
1753 };
1754
1755 extern void endutent(void);
1756 extern struct utmp *getutent(void);
1757 extern void setutent(void);
1758 extern int getutent_r(struct utmp *, struct utmp **);
1759 extern int utmpname(const char *);
1760 extern int login_tty(int);
1761 extern void login(const struct utmp *);
1762 extern int logout(const char *);
1763 extern void logwtmp(const char *, const char *, const char *);

```

11.3.2377 utmpx.h

```

1764
1765 struct utmpx {
1766     †
1767     short ut_type;
1768     pid_t ut_pid;
1769     char ut_line[UT_LINESIZE];
1770     char ut_id[4];
1771     char ut_user[UT_NAMESIZE];
1772     char ut_host[UT_HOSTSIZE];
1773     struct exit_status ut_exit;
1774     long int ut_session;
1775     struct timeval ut_tv;
1776     int32_t ut_addr_v6[4];
1777     char __unused[20];
1778 };
1779

```

```

1780     extern void endutxent(void);
1781     extern struct utmpx *getutxent(void);
1782     extern struct utmpx *getutxid(const struct utmpx *);
1783     extern struct utmpx *getutxline(const struct utmpx *);
1784     extern struct utmpx *pututxline(const struct utmpx *);
1785     extern void setutxent(void);

```

11.3.78 wchar.h

```

1786
1787     extern double __wcstod_internal(const wchar_t *, wchar_t **, int);
1788     extern float __wcstof_internal(const wchar_t *, wchar_t **, int);
1789     extern long int __wcstol_internal(const wchar_t *, wchar_t **, int,
1790     int);
1791     extern long double __wcstold_internal(const wchar_t *, wchar_t **, int);
1792     extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t *
1793     *,
1794     int, int);
1795     extern wchar_t *wscat(wchar_t *, const wchar_t *);
1796     extern wchar_t *wcschr(const wchar_t *, wchar_t);
1797     extern int wcsncmp(const wchar_t *, const wchar_t *);
1798     extern int wscoll(const wchar_t *, const wchar_t *);
1799     extern wchar_t *wcscpy(wchar_t *, const wchar_t *);
1800     extern size_t wcsncpy(const wchar_t *, const wchar_t *);
1801     extern wchar_t *wcsdup(const wchar_t *);
1802     extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
1803     extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
1804     extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
1805     extern wchar_t *wcpbrk(const wchar_t *, const wchar_t *);
1806     extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
1807     extern size_t wcsspn(const wchar_t *, const wchar_t *);
1808     extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
1809     extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t **);
1810     extern int wcswidth(const wchar_t *, size_t);
1811     extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
1812     extern int wctob(wint_t);
1813     extern int wcwidth(wchar_t);
1814     extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
1815     extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
1816     extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
1817     extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
1818     extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
1819     extern size_t mbrlen(const char *, size_t, mbstate_t *);
1820     extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
1821     extern int mbsinit(const mbstate_t *);
1822     extern size_t mbsnrtowcs(wchar_t *, const char **, size_t, size_t,
1823     mbstate_t *);
1824     extern size_t mbsrtowcs(wchar_t *, const char **, size_t, mbstate_t *);
1825     extern wchar_t *wcpncpy(wchar_t *, const wchar_t *);
1826     extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
1827     extern size_t wctomb(char *, wchar_t, mbstate_t *);
1828     extern size_t wcslen(const wchar_t *);
1829     extern size_t wcsnrtombs(char *, const wchar_t **, size_t, size_t,
1830     mbstate_t *);
1831     extern size_t wcsrtombs(char *, const wchar_t **, size_t, mbstate_t *);
1832     extern double wcstod(const wchar_t *, wchar_t **);
1833     extern float wcstof(const wchar_t *, wchar_t **);
1834     extern long int wcstol(const wchar_t *, wchar_t **, int);
1835     extern long double wcstold(const wchar_t *, wchar_t **);
1836     extern long long int wcstoll(const wchar_t *, wchar_t **, int);
1837     extern unsigned long int wcstoul(const wchar_t *, wchar_t **, int);
1838     extern unsigned long long int wcstoull(const wchar_t *, wchar_t **, int);
1839     extern wchar_t *wswcs(const wchar_t *, const wchar_t *);
1840     extern int wscasecmp(const wchar_t *, const wchar_t *);

```



```

1841 extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
1842 extern size_t wcsnlen(const wchar_t *, size_t);
1843 extern long long int wcstoll(const wchar_t *, wchar_t ** , int);
1844 extern unsigned long long int wcstoull(const wchar_t *, wchar_t ** , int);
1845 extern wint_t btowc(int);
1846 extern wint_t fgetwc(FILE *);
1847 extern wint_t fgetwc_unlocked(FILE *);
1848 extern wchar_t *fgetws(wchar_t *, int, FILE *);
1849 extern wint_t fputwc(wchar_t, FILE *);
1850 extern int fputws(const wchar_t *, FILE *);
1851 extern int fwide(FILE *, int);
1852 extern int fwprintf(FILE *, const wchar_t *, ...);
1853 extern int fwscanf(FILE *, const wchar_t *, ...);
1854 extern wint_t getwc(FILE *);
1855 extern wint_t getwchar(void);
1856 extern wint_t putwc(wchar_t, FILE *);
1857 extern wint_t putwchar(wchar_t);
1858 extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
1859 extern int swscanf(const wchar_t *, const wchar_t *, ...);
1860 extern wint_t ungetwc(wint_t, FILE *);
1861 extern int vfwprintf(FILE *, const wchar_t *, va_list);
1862 extern int vfwscanf(FILE *, const wchar_t *, va_list);
1863 extern int vswprintf(wchar_t *, size_t, const wchar_t *, va_list);
1864 extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
1865 extern int vwprintf(const wchar_t *, va_list);
1866 extern int vwscanf(const wchar_t *, va_list);
1867 extern size_t wcsftime(wchar_t *, size_t, const wchar_t *,
1868                       const struct tm *);
1869 extern int wprintf(const wchar_t *, ...);
1870 extern int wscanf(const wchar_t *, ...);

```

11.3.79 wctype.h

```

1871
1872 extern int iswblank(wint_t);
1873 extern wint_t towlower(wint_t);
1874 extern wint_t towupper(wint_t);
1875 extern wctrans_t wctrans(const char *);
1876 extern int iswalnum(wint_t);
1877 extern int iswalphabetic(wint_t);
1878 extern int iswcntrl(wint_t);
1879 extern int iswctype(wint_t, wctype_t);
1880 extern int iswdigit(wint_t);
1881 extern int iswgraph(wint_t);
1882 extern int iswlower(wint_t);
1883 extern int iswprint(wint_t);
1884 extern int iswpunct(wint_t);
1885 extern int iswspace(wint_t);
1886 extern int iswupper(wint_t);
1887 extern int iswxdigit(wint_t);
1888 extern wctype_t wctype(const char *);
1889 extern wint_t towctrans(wint_t, wctrans_t);

```

11.3.80 wordexp.h

```

1890
1891 extern int wordexp(const char *, wordexp_t *, int);
1892 extern void wordfree(wordexp_t *);

```

11.4 Interfaces for libm

1893 Table 11-24 defines the library name and shared object name for the libm library

1894

Table 11-24 libm Definition

Library:	libm
SONAME:	libm.so.6

1895

1896

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

1897

[ISO C99] ISO C (1999)
 [LSB] ~~this specification~~ This Specification
 [SUSv2] SUSv2
 [SUSv3] ISO POSIX (2003)

1898

11.4.1 Math

1899

11.4.1.1 Interfaces for Math

1900

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.

1901

1902

1903

Table 11-25 libm - Math Function Interfaces

__finite(GLIBC_2.1) [1]	ceilf(GLIBC_2.0) [2]	exp2f(GLIBC_2.1) [2]	jnf(GLIBC_2.0) [1]	remainder(GLIBC_2.0) [2]
__finitef(GLIBC_2.1) [1]	ceil(GLIBC_2.0) [2]	expf(GLIBC_2.0) [2]	jnl(GLIBC_2.0) [1]	remainderf(GLIBC_2.0) [2]
__finitel(GLIBC_2.1) [1]	exp(GLIBC_2.1) [2]	expl(GLIBC_2.0) [2]	ldexp(GLIBC_2.0) [2]	remainderl(GLIBC_2.0) [2]
__fpclassify(GLIBC_2.1) [3]	expf(GLIBC_2.1) [2]	expm1(GLIBC_2.0) [2]	ldexpf(GLIBC_2.0) [2]	remquo(GLIBC_2.1) [2]
__fpclassifyf(GLIBC_2.1) [3]	expl(GLIBC_2.1) [2]	expm1f(GLIBC_2.0) [2]	ldexpl(GLIBC_2.0) [2]	remquof(GLIBC_2.1) [2]
acos(GLIBC_2.0) [2]	eimag(GLIBC_2.1) [2]	expm1l(GLIBC_2.0) [2]	lgamma(GLIBC_2.0) [2]	remquol(GLIBC_2.1) [2]
acosf(GLIBC_2.0) [2]	eimagf(GLIBC_2.1) [2]	fabs(GLIBC_2.0) [2]	lgamma_r(GLIBC_2.0) [1]	rint(GLIBC_2.0) [2]
acosl(GLIBC_2.0) [2]	eimagl(GLIBC_2.1) [2]	fabsf(GLIBC_2.0) [2]	lgammaf(GLIBC_2.0) [2]	rintf(GLIBC_2.0) [2]
acosh(GLIBC_2.0) [2]	elog(GLIBC_2.1) [2]	fabsl(GLIBC_2.0) [2]	lgammaf_r(GLIBC_2.0) [1]	rintl(GLIBC_2.0) [2]
acoshf(GLIBC_2.0) [2]	elog10(GLIBC_2.1) [1]	fdim(GLIBC_2.1) [2]	lgammal(GLIBC_2.0) [2]	round(GLIBC_2.1) [2]
acosl(GLIBC_2.0) [2]	elog10f(GLIBC_2.1) [1]	fdimf(GLIBC_2.1) [2]	lgammal_r(GLIBC_2.0) [1]	roundf(GLIBC_2.1) [2]
asin(GLIBC_2.0) [2]	elog10l(GLIBC_2.1) [1]	fdiml(GLIBC_2.1) [2]	llrint(GLIBC_2.1) [2]	roundl(GLIBC_2.1) [2]

<code>asinh(GLIBC_2.0)</code> [2]	<code>elogf(GLIBC_2.1)</code> [2]	<code>felearexcept(GLIBC_2.1)</code> [2]	<code>llrintf(GLIBC_2.1)</code> [2]	<code>scalb(GLIBC_2.0)</code> [2]
<code>asinh(GLIBC_2.0)</code> [2]	<code>elogl(GLIBC_2.1)</code> [2]	<code>fegetenv(GLIBC_2.1)</code> [2]	<code>llrintl(GLIBC_2.1)</code> [2]	<code>scalbf(GLIBC_2.0)</code> [1]
<code>asinhf(GLIBC_2.0)</code> [2]	<code>conj(GLIBC_2.1)</code> [2]	<code>fegetexceptflag(GLIBC_2.1)</code> [2]	<code>llround(GLIBC_2.1)</code> [2]	<code>scalbl(GLIBC_2.0)</code> [1]
<code>asinhf(GLIBC_2.0)</code> [2]	<code>conjf(GLIBC_2.1)</code> [2]	<code>fegetround(GLIBC_2.1)</code> [2]	<code>llroundf(GLIBC_2.1)</code> [2]	<code>scalbln(GLIBC_2.1)</code> [2]
<code>asinl(GLIBC_2.0)</code> [2]	<code>conjl(GLIBC_2.1)</code> [2]	<code>feholdexcept(GLIBC_2.1)</code> [2]	<code>llroundl(GLIBC_2.1)</code> [2]	<code>scalblnf(GLIBC_2.1)</code> [2]
<code>atan(GLIBC_2.0)</code> [2]	<code>copysign(GLIBC_2.0)</code> [2]	<code>feraiseexcept(GLIBC_2.1)</code> [2]	<code>log(GLIBC_2.0)</code> [2]	<code>scalbln(GLIBC_2.1)</code> [2]
<code>atan2(GLIBC_2.0)</code> [2]	<code>copysignf(GLIBC_2.0)</code> [2]	<code>fesetenv(GLIBC_2.1)</code> [2]	<code>log10(GLIBC_2.0)</code> [2]	<code>scalbn(GLIBC_2.0)</code> [2]
<code>atan2f(GLIBC_2.0)</code> [2]	<code>copysignl(GLIBC_2.0)</code> [2]	<code>fesetexceptflag(GLIBC_2.1)</code> [2]	<code>log10f(GLIBC_2.0)</code> [2]	<code>scalbnf(GLIBC_2.0)</code> [2]
<code>atan2l(GLIBC_2.0)</code> [2]	<code>cos(GLIBC_2.0)</code> [2]	<code>fesetround(GLIBC_2.1)</code> [2]	<code>log10l(GLIBC_2.0)</code> [2]	<code>scalbnl(GLIBC_2.0)</code> [2]
<code>atanf(GLIBC_2.0)</code> [2]	<code>cosf(GLIBC_2.0)</code> [2]	<code>fetestexcept(GLIBC_2.1)</code> [2]	<code>log1p(GLIBC_2.0)</code> [2]	<code>significand(GLIBC_2.0)</code> [1]
<code>atanh(GLIBC_2.0)</code> [2]	<code>cosh(GLIBC_2.0)</code> [2]	<code>feupdateenv(GLIBC_2.1)</code> [2]	<code>log1pf(GLIBC_2.0)</code> [2]	<code>significandf(GLIBC_2.0)</code> [1]
<code>atanhf(GLIBC_2.0)</code> [2]	<code>coshf(GLIBC_2.0)</code> [2]	<code>finite(GLIBC_2.0)</code> [4]	<code>log1pl(GLIBC_2.0)</code> [2]	<code>significandl(GLIBC_2.0)</code> [1]
<code>atanhl(GLIBC_2.0)</code> [2]	<code>coshl(GLIBC_2.0)</code> [2]	<code>finitef(GLIBC_2.0)</code> [1]	<code>log2(GLIBC_2.1)</code> [2]	<code>sin(GLIBC_2.0)</code> [2]
<code>atanl(GLIBC_2.0)</code> [2]	<code>cosl(GLIBC_2.0)</code> [2]	<code>finitel(GLIBC_2.0)</code> [1]	<code>log2f(GLIBC_2.1)</code> [2]	<code>sincos(GLIBC_2.1)</code> [1]
<code>eabs(GLIBC_2.1)</code> [2]	<code>epow(GLIBC_2.1)</code> [2]	<code>floor(GLIBC_2.0)</code> [2]	<code>log2l(GLIBC_2.1)</code> [2]	<code>sincosf(GLIBC_2.1)</code> [1]
<code>eabsf(GLIBC_2.1)</code> [2]	<code>epowf(GLIBC_2.1)</code> [2]	<code>floorf(GLIBC_2.0)</code> [2]	<code>logb(GLIBC_2.0)</code> [2]	<code>sincosl(GLIBC_2.1)</code> [1]
<code>eabsl(GLIBC_2.1)</code> [2]	<code>epowl(GLIBC_2.1)</code> [2]	<code>floorl(GLIBC_2.0)</code> [2]	<code>logbf(GLIBC_2.0)</code> [2]	<code>sinf(GLIBC_2.0)</code> [2]
<code>eacos(GLIBC_2.1)</code> [2]	<code>eprojl(GLIBC_2.1)</code> [2]	<code>fma(GLIBC_2.1)</code> [2]	<code>logbl(GLIBC_2.0)</code> [2]	<code>sinh(GLIBC_2.0)</code> [2]

<code>caacosf(GLIBC_2.1)</code> [2]	<code>eprojf(GLIBC_2.1)</code> [2]	<code>fmaf(GLIBC_2.1)</code> [2]	<code>logf(GLIBC_2.0)</code> [2]	<code>sinhf(GLIBC_2.0)</code> [2]
<code>caacosh(GLIBC_2.1)</code> [2]	<code>eprojl(GLIBC_2.1)</code> [2]	<code>fmal(GLIBC_2.1)</code> [2]	<code>logl(GLIBC_2.0)</code> [2]	<code>sinhl(GLIBC_2.0)</code> [2]
<code>caacoshf(GLIBC_2.1)</code> [2]	<code>erealf(GLIBC_2.1)</code> [2]	<code>fmaxf(GLIBC_2.1)</code> [2]	<code>lrintf(GLIBC_2.1)</code> [2]	<code>sinl(GLIBC_2.0)</code> [2]
<code>caacoshl(GLIBC_2.1)</code> [2]	<code>ereall(GLIBC_2.1)</code> [2]	<code>fmaxl(GLIBC_2.1)</code> [2]	<code>lrintf(GLIBC_2.1)</code> [2]	<code>sqrt(GLIBC_2.0)</code> [2]
<code>caacosl(GLIBC_2.1)</code> [2]	<code>ereall(GLIBC_2.1)</code> [2]	<code>fmaxl(GLIBC_2.1)</code> [2]	<code>lrintl(GLIBC_2.1)</code> [2]	<code>sqrtf(GLIBC_2.0)</code> [2]
<code>carg(GLIBC_2.1)</code> [2]	<code>esin(GLIBC_2.1)</code> [2]	<code>fmin(GLIBC_2.1)</code> [2]	<code>lround(GLIBC_2.1)</code> [2]	<code>sqrtl(GLIBC_2.0)</code> [2]
<code>cargf(GLIBC_2.1)</code> [2]	<code>esinf(GLIBC_2.1)</code> [2]	<code>fminf(GLIBC_2.1)</code> [2]	<code>lroundf(GLIBC_2.1)</code> [2]	<code>tan(GLIBC_2.0)</code> [2]
<code>cargl(GLIBC_2.1)</code> [2]	<code>esinh(GLIBC_2.1)</code> [2]	<code>fminl(GLIBC_2.1)</code> [2]	<code>lroundl(GLIBC_2.1)</code> [2]	<code>tanf(GLIBC_2.0)</code> [2]
<code>casin(GLIBC_2.1)</code> [2]	<code>esinhf(GLIBC_2.1)</code> [2]	<code>fmod(GLIBC_2.0)</code> [2]	<code>matherr(GLIBC_2.0)</code> [1]	<code>tanh(GLIBC_2.0)</code> [2]
<code>casinf(GLIBC_2.1)</code> [2]	<code>esinhl(GLIBC_2.1)</code> [2]	<code>fmodf(GLIBC_2.0)</code> [2]	<code>modf(GLIBC_2.0)</code> [2]	<code>tanhf(GLIBC_2.0)</code> [2]
<code>casinh(GLIBC_2.1)</code> [2]	<code>esinl(GLIBC_2.1)</code> [2]	<code>fmodl(GLIBC_2.0)</code> [2]	<code>modff(GLIBC_2.0)</code> [2]	<code>tanhf(GLIBC_2.0)</code> [2]
<code>casinhf(GLIBC_2.1)</code> [2]	<code>esqrt(GLIBC_2.1)</code> [2]	<code>frexp(GLIBC_2.0)</code> [2]	<code>modfl(GLIBC_2.0)</code> [2]	<code>tanl(GLIBC_2.0)</code> [2]
<code>casinhl(GLIBC_2.1)</code> [2]	<code>esqrtf(GLIBC_2.1)</code> [2]	<code>frexpf(GLIBC_2.0)</code> [2]	<code>nan(GLIBC_2.1)</code> [2]	<code>tgamma(GLIBC_2.1)</code> [2]
<code>casinl(GLIBC_2.1)</code> [2]	<code>esqrtl(GLIBC_2.1)</code> [2]	<code>frexpl(GLIBC_2.0)</code> [2]	<code>nanf(GLIBC_2.1)</code> [2]	<code>tgammaf(GLIBC_2.1)</code> [2]
<code>catan(GLIBC_2.1)</code> [2]	<code>etan(GLIBC_2.1)</code> [2]	<code>gamma(GLIBC_2.0)</code> [4]	<code>nanl(GLIBC_2.1)</code> [2]	<code>tgammal(GLIBC_2.1)</code> [2]
<code>catanf(GLIBC_2.1)</code> [2]	<code>etanf(GLIBC_2.1)</code> [2]	<code>gammaf(GLIBC_2.0)</code> [1]	<code>nearbyint(GLIBC_2.1)</code> [2]	<code>trunc(GLIBC_2.1)</code> [2]
<code>catanh(GLIBC_2.1)</code> [2]	<code>etanh(GLIBC_2.1)</code> [2]	<code>gammal(GLIBC_2.0)</code> [1]	<code>nearbyintf(GLIBC_2.1)</code> [2]	<code>truncf(GLIBC_2.1)</code> [2]
<code>catanhf(GLIBC_2.1)</code> [2]	<code>etanhf(GLIBC_2.1)</code> [2]	<code>hypot(GLIBC_2.0)</code> [2]	<code>nearbyintl(GLIBC_2.1)</code> [2]	<code>truncl(GLIBC_2.1)</code> [2]
<code>catanhl(GLIBC_2.1)</code> [2]	<code>etanhl(GLIBC_2.1)</code> [2]	<code>hypotf(GLIBC_2.0)</code> [2]	<code>nextafter(GLIBC_2.0)</code> [2]	<code>y0(GLIBC_2.0)</code> [2]
<code>catanl(GLIBC_2.1)</code> [2]	<code>etanl(GLIBC_2.1)</code> [2]	<code>hypotl(GLIBC_2.0)</code> [2]	<code>nextafterf(GLIBC_2.0)</code> [2]	<code>y0f(GLIBC_2.0)</code> [1]
<code>cbrrt(GLIBC_2.1)</code> [2]	<code>dremf(GLIBC_2.1)</code> [2]	<code>ilogb(GLIBC_2.1)</code> [2]	<code>nextafterl(GLIBC_2.0)</code> [2]	<code>y0l(GLIBC_2.0)</code> [2]

0) [2]	-2.0) [1]	2.0) [2]	BC_2.0) [2]	0) [1]
cbtrf(GLIBC_2.0) [2]	dreml(GLIBC_2.0) [1]	ilogbf(GLIBC_2.0) [2]	nexttoward(GLIBC_2.1) [2]	y1(GLIBC_2.0) [2]
cbtrl(GLIBC_2.0) [2]	erf(GLIBC_2.0) [2]	ilogbl(GLIBC_2.0) [2]	nexttowardf(GLIBC_2.1) [2]	y1f(GLIBC_2.0) [1]
ceos(GLIBC_2.1) [2]	erfe(GLIBC_2.0) [2]	j0(GLIBC_2.0) [2]	nexttowardl(GLIBC_2.1) [2]	y1l(GLIBC_2.0) [1]
ceosf(GLIBC_2.1) [2]	erfef(GLIBC_2.0) [2]	j0f(GLIBC_2.0) [1]	pow(GLIBC_2.0) [2]	yn(GLIBC_2.0) [2]
ceosh(GLIBC_2.1) [2]	erfel(GLIBC_2.0) [2]	j0l(GLIBC_2.0) [1]	pow10(GLIBC_2.1) [1]	ynf(GLIBC_2.0) [1]
ceoshf(GLIBC_2.1) [2]	erfff(GLIBC_2.0) [2]	j1(GLIBC_2.0) [2]	pow10f(GLIBC_2.1) [1]	ynl(GLIBC_2.0) [1]
ceoshl(GLIBC_2.1) [2]	erfl(GLIBC_2.0) [2]	j1f(GLIBC_2.0) [1]	pow10l(GLIBC_2.1) [1]	
ceosl(GLIBC_2.1) [2]	exp(GLIBC_2.0) [2]	j1l(GLIBC_2.0) [1]	powf(GLIBC_2.0) [2]	
ceil(GLIBC_2.0) [2]	exp2(GLIBC_2.1) [2]	jn(GLIBC_2.0) [2]	powl(GLIBC_2.0) [2]	

1904

1905

1906

*Referenced Specification(s)***[1]**

__finite(GLIBC_2.1) [ISO C99]	__finitef(GLIBC_2.1) [ISO C99]	__finitel(GLIBC_2.1) [ISO C99]	__fpclassify(GLIBC_2.1) [LSB]
__fpclassifyf(GLIBC_2.1) [LSB]	acos(GLIBC_2.0) [SUSv3]	acosf(GLIBC_2.0) [SUSv3]	acosh(GLIBC_2.0) [SUSv3]
acoshf(GLIBC_2.0) [SUSv3]	acoshl(GLIBC_2.0) [SUSv3]	acosl(GLIBC_2.0) [SUSv3]	asin(GLIBC_2.0) [SUSv3]
asinf(GLIBC_2.0) [SUSv3]	asinh(GLIBC_2.0) [SUSv3]	asinhf(GLIBC_2.0) [SUSv3]	asinhf(GLIBC_2.0) [SUSv3]
asinl(GLIBC_2.0) [SUSv3]	atan(GLIBC_2.0) [SUSv3]	atan2(GLIBC_2.0) [SUSv3]	atan2f(GLIBC_2.0) [SUSv3]
atan2l(GLIBC_2.0) [SUSv3]	atanf(GLIBC_2.0) [SUSv3]	atanh(GLIBC_2.0) [SUSv3]	atanhf(GLIBC_2.0) [SUSv3]
atanhl(GLIBC_2.0) [SUSv3]	atanl(GLIBC_2.0) [SUSv3]	cabs(GLIBC_2.1) [SUSv3]	cabsf(GLIBC_2.1) [SUSv3]
cabsl(GLIBC_2.1) [SUSv3]	cacos(GLIBC_2.1) [SUSv3]	cacosf(GLIBC_2.1) [SUSv3]	cacosh(GLIBC_2.1) [SUSv3]
cacoshf(GLIBC_2.1) [SUSv3]	cacoshl(GLIBC_2.1) [SUSv3]	cacosl(GLIBC_2.1) [SUSv3]	carg(GLIBC_2.1) [SUSv3]

cargf(GLIBC_2.1) [SUSv3]	cargl(GLIBC_2.1) [SUSv3]	casin(GLIBC_2.1) [SUSv3]	casinf(GLIBC_2.1) [SUSv3]
casinh(GLIBC_2.1) [SUSv3]	casinhf(GLIBC_2.1) [SUSv3]	casinhl(GLIBC_2.1) [SUSv3]	casinl(GLIBC_2.1) [SUSv3]
catan(GLIBC_2.1) [SUSv3]	catanf(GLIBC_2.1) [SUSv3]	catanh(GLIBC_2.1) [SUSv3]	catanhf(GLIBC_2.1) [SUSv3]
catanhl(GLIBC_2.1) [SUSv3]	catanl(GLIBC_2.1) [SUSv3]	cbrt(GLIBC_2.0) [SUSv3]	cbrtf(GLIBC_2.0) [SUSv3]
cbrtl(GLIBC_2.0) [SUSv3]	ccos(GLIBC_2.1) [SUSv3]	ccosf(GLIBC_2.1) [SUSv3]	ccosh(GLIBC_2.1) [SUSv3]
ccoshf(GLIBC_2.1) [SUSv3]	ccoshl(GLIBC_2.1) [SUSv3]	ccosl(GLIBC_2.1) [SUSv3]	ceil(GLIBC_2.0) [SUSv3]
ceilf(GLIBC_2.0) [SUSv3]	ceill(GLIBC_2.0) [SUSv3]	cexp(GLIBC_2.1) [SUSv3]	cexpf(GLIBC_2.1) [SUSv3]
cexpl(GLIBC_2.1) [SUSv3]	cimag(GLIBC_2.1) [SUSv3]	cimagf(GLIBC_2.1) [SUSv3]	cimagl(GLIBC_2.1) [SUSv3]
clog(GLIBC_2.1) [SUSv3]	clog10(GLIBC_2.1) [ISOC99]	clog10f(GLIBC_2.1) [ISOC99]	clog10l(GLIBC_2.1) [ISOC99]
clogf(GLIBC_2.1) [SUSv3]	clogl(GLIBC_2.1) [SUSv3]	conj(GLIBC_2.1) [SUSv3]	conjf(GLIBC_2.1) [SUSv3]
conjl(GLIBC_2.1) [SUSv3]	copysign(GLIBC_2.0) [SUSv3]	copysignf(GLIBC_2.0) [SUSv3]	copysignl(GLIBC_2.0) [SUSv3]
cos(GLIBC_2.0) [SUSv3]	cosf(GLIBC_2.0) [SUSv3]	cosh(GLIBC_2.0) [SUSv3]	coshf(GLIBC_2.0) [SUSv3]
coshl(GLIBC_2.0) [SUSv3]	cosl(GLIBC_2.0) [SUSv3]	cpow(GLIBC_2.1) [SUSv3]	cpowf(GLIBC_2.1) [SUSv3]
cpowl(GLIBC_2.1) [SUSv3]	cproj(GLIBC_2.1) [SUSv3]	cprojf(GLIBC_2.1) [SUSv3]	cprojl(GLIBC_2.1) [SUSv3]
creal(GLIBC_2.1) [SUSv3]	crealf(GLIBC_2.1) [SUSv3]	creall(GLIBC_2.1) [SUSv3]	csin(GLIBC_2.1) [SUSv3]
csinf(GLIBC_2.1) [SUSv3]	csinh(GLIBC_2.1) [SUSv3]	csinhf(GLIBC_2.1) [SUSv3]	csinhl(GLIBC_2.1) [SUSv3]
csinl(GLIBC_2.1) [SUSv3]	csqrt(GLIBC_2.1) [SUSv3]	csqrtf(GLIBC_2.1) [SUSv3]	csqrtl(GLIBC_2.1) [SUSv3]
ctan(GLIBC_2.1) [SUSv3]	ctanf(GLIBC_2.1) [SUSv3]	ctanh(GLIBC_2.1) [SUSv3]	ctanhf(GLIBC_2.1) [SUSv3]
ctanhl(GLIBC_2.1) [SUSv3]	ctanl(GLIBC_2.1) [SUSv3]	dremf(GLIBC_2.0) [ISOC99]	dreml(GLIBC_2.0) [ISOC99]
erf(GLIBC_2.0) [SUSv3]	erfc(GLIBC_2.0) [SUSv3]	erfcf(GLIBC_2.0) [SUSv3]	erfcl(GLIBC_2.0) [SUSv3]
erff(GLIBC_2.0)	erfl(GLIBC_2.0)	exp(GLIBC_2.0)	exp2(GLIBC_2.1)

[SUSv3]	[SUSv3]	[SUSv3]	[SUSv3]
exp2f(GLIBC_2.1) [SUSv3]	expf(GLIBC_2.0) [SUSv3]	expl(GLIBC_2.0) [SUSv3]	expm1(GLIBC_2.0) [SUSv3]
expm1f(GLIBC_2.0) [SUSv3]	expm1l(GLIBC_2.0) [SUSv3]	fabs(GLIBC_2.0) [SUSv3]	fabsf(GLIBC_2.0) [SUSv3]
fabsl(GLIBC_2.0) [SUSv3]	fdim(GLIBC_2.1) [SUSv3]	fdimf(GLIBC_2.1) [SUSv3]	fdiml(GLIBC_2.1) [SUSv3]
feclearexcept(GLIBC_2.1) [SUSv3]	fegetenv(GLIBC_2.1) [SUSv3]	fegetexceptflag(GLIBC_2.1) [SUSv3]	fegetround(GLIBC_2.1) [SUSv3]
feholdexcept(GLIBC_2.1) [SUSv3]	feraiseexcept(GLIBC_2.1) [SUSv3]	fesetenv(GLIBC_2.1) [SUSv3]	fesetexceptflag(GLIBC_2.1) [SUSv3]
fesetround(GLIBC_2.1) [SUSv3]	fetestexcept(GLIBC_2.1) [SUSv3]	feupdateenv(GLIBC_2.1) [SUSv3]	finite(GLIBC_2.0) [SUSv2]
finitef(GLIBC_2.0) [ISOC99]	finitel(GLIBC_2.0) [ISOC99]	floor(GLIBC_2.0) [SUSv3]	floorf(GLIBC_2.0) [SUSv3]
floorl(GLIBC_2.0) [SUSv3]	fma(GLIBC_2.1) [SUSv3]	fmaf(GLIBC_2.1) [SUSv3]	fmal(GLIBC_2.1) [SUSv3]
fmax(GLIBC_2.1) [SUSv3]	fmaxf(GLIBC_2.1) [SUSv3]	fmaxl(GLIBC_2.1) [SUSv3]	fmin(GLIBC_2.1) [SUSv3]
fminf(GLIBC_2.1) [SUSv3]	fminl(GLIBC_2.1) [SUSv3]	fmod(GLIBC_2.0) [SUSv3]	fmodf(GLIBC_2.0) [SUSv3]
fmodl(GLIBC_2.0) [SUSv3]	frexp(GLIBC_2.0) [SUSv3]	frexpf(GLIBC_2.0) [SUSv3]	frexpl(GLIBC_2.0) [SUSv3]
gamma(GLIBC_2.0) [SUSv2]	gammaf(GLIBC_2.0) [ISOC99]	gammal(GLIBC_2.0) [ISOC99]	hypot(GLIBC_2.0) [SUSv3]
hypotf(GLIBC_2.0) [SUSv3]	hypotl(GLIBC_2.0) [SUSv3]	ilogb(GLIBC_2.0) [SUSv3]	ilogbf(GLIBC_2.0) [SUSv3]
ilogbl(GLIBC_2.0) [SUSv3]	j0(GLIBC_2.0) [SUSv3]	j0f(GLIBC_2.0) [ISOC99]	j0l(GLIBC_2.0) [ISOC99]
j1(GLIBC_2.0) [SUSv3]	j1f(GLIBC_2.0) [ISOC99]	j1l(GLIBC_2.0) [ISOC99]	jnl(GLIBC_2.0) [SUSv3]
jnf(GLIBC_2.0) [ISOC99]	jnl(GLIBC_2.0) [ISOC99]	ldexp(GLIBC_2.0) [SUSv3]	ldexpf(GLIBC_2.0) [SUSv3]
ldexpl(GLIBC_2.0) [SUSv3]	lgamma(GLIBC_2.0) [SUSv3]	lgamma_r(GLIBC_2.0) [ISOC99]	lgammaf(GLIBC_2.0) [SUSv3]
lgammaf_r(GLIBC_2.0) [ISOC99]	lgammal(GLIBC_2.0) [SUSv3]	lgammal_r(GLIBC_2.0) [ISOC99]	llrint(GLIBC_2.1) [SUSv3]
llrintf(GLIBC_2.1) [SUSv3]	llrintl(GLIBC_2.1) [SUSv3]	llround(GLIBC_2.1) [SUSv3]	llroundf(GLIBC_2.1) [SUSv3]
llroundl(GLIBC_2.1) [SUSv3]	log(GLIBC_2.0) [SUSv3]	log10(GLIBC_2.0) [SUSv3]	log10f(GLIBC_2.0) [SUSv3]

log10l(GLIBC_2.0) [SUSv3]	log1p(GLIBC_2.0) [SUSv3]	log1pf(GLIBC_2.0) [SUSv3]	log1pl(GLIBC_2.0) [SUSv3]
log2(GLIBC_2.1) [SUSv3]	log2f(GLIBC_2.1) [SUSv3]	log2l(GLIBC_2.1) [SUSv3]	logb(GLIBC_2.0) [SUSv3]
logbf(GLIBC_2.0) [SUSv3]	logbl(GLIBC_2.0) [SUSv3]	logf(GLIBC_2.0) [SUSv3]	logl(GLIBC_2.0) [SUSv3]
lrint(GLIBC_2.1) [SUSv3]	lrintf(GLIBC_2.1) [SUSv3]	lrintl(GLIBC_2.1) [SUSv3]	lround(GLIBC_2.1) [SUSv3]
lroundf(GLIBC_2.1) [SUSv3]	lroundl(GLIBC_2.1) [SUSv3]	matherr(GLIBC_2.0) [ISOC99]	modf(GLIBC_2.0) [SUSv3]
modff(GLIBC_2.0) [SUSv3]	modfl(GLIBC_2.0) [SUSv3]	nan(GLIBC_2.1) [SUSv3]	nanf(GLIBC_2.1) [SUSv3]
nanl(GLIBC_2.1) [SUSv3]	nearbyint(GLIBC_2.1) [SUSv3]	nearbyintf(GLIBC_2.1) [SUSv3]	nearbyintl(GLIBC_2.1) [SUSv3]
nextafter(GLIBC_2.0) [SUSv3]	nextafterf(GLIBC_2.0) [SUSv3]	nextafterl(GLIBC_2.0) [SUSv3]	nexttoward(GLIBC_2.1) [SUSv3]
nexttowardf(GLIBC_2.1) [SUSv3]	nexttowardl(GLIBC_2.1) [SUSv3]	pow(GLIBC_2.0) [SUSv3]	pow10(GLIBC_2.1) [ISOC99]
pow10f(GLIBC_2.1) [ISOC99]	pow10l(GLIBC_2.1) [ISOC99]	powf(GLIBC_2.0) [SUSv3]	powl(GLIBC_2.0) [SUSv3]
remainder(GLIBC_2.0) [SUSv3]	remainderf(GLIBC_2.0) [SUSv3]	remainderl(GLIBC_2.0) [SUSv3]	remquo(GLIBC_2.1) [SUSv3]
remquof(GLIBC_2.1) [SUSv3]	remquol(GLIBC_2.1) [SUSv3]	rint(GLIBC_2.0) [SUSv3]	rintf(GLIBC_2.0) [SUSv3]
rintl(GLIBC_2.0) [SUSv3]	round(GLIBC_2.1) [SUSv3]	roundf(GLIBC_2.1) [SUSv3]	roundl(GLIBC_2.1) [SUSv3]
scalb(GLIBC_2.0) [SUSv3]	scalbf(GLIBC_2.0) [ISOC99]	scalbl(GLIBC_2.0) [ISOC99]	scalbln(GLIBC_2.1) [SUSv3]
scalblnf(GLIBC_2.1) [SUSv3]	scalblnl(GLIBC_2.1) [SUSv3]	scalbn(GLIBC_2.0) [SUSv3]	scalbnf(GLIBC_2.0) [SUSv3]
scalbnl(GLIBC_2.0) [SUSv3]	significand(GLIBC_2.0) [ISOC99]	significandf(GLIBC_2.0) [ISOC99]	significandl(GLIBC_2.0) [ISOC99]
sin(GLIBC_2.0) [SUSv3]	sincos(GLIBC_2.1) [ISOC99]	sincosf(GLIBC_2.1) [ISOC99]	sincosl(GLIBC_2.1) [ISOC99]
sinf(GLIBC_2.0) [SUSv3]	sinh(GLIBC_2.0) [SUSv3]	sinhf(GLIBC_2.0) [SUSv3]	sinhl(GLIBC_2.0) [SUSv3]
sinl(GLIBC_2.0) [SUSv3]	sqrt(GLIBC_2.0) [SUSv3]	sqrtf(GLIBC_2.0) [SUSv3]	sqrtl(GLIBC_2.0) [SUSv3]
tan(GLIBC_2.0) [SUSv3]	tanf(GLIBC_2.0) [SUSv3]	tanh(GLIBC_2.0) [SUSv3]	tanhf(GLIBC_2.0) [SUSv3]
tanh(GLIBC_2.0)	tanl(GLIBC_2.0)	tgamma(GLIBC_2)	tgammaf(GLIBC_2)

[SUSv3]	[SUSv3]	.1) [SUSv3]	2.1) [SUSv3]
tgammal(GLIBC_2.1) [SUSv3]	trunc(GLIBC_2.1) [SUSv3]	truncf(GLIBC_2.1) [SUSv3]	truncl(GLIBC_2.1) [SUSv3]
y0(GLIBC_2.0) [SUSv3]	y0f(GLIBC_2.0) [ISOC99]	y0l(GLIBC_2.0) [ISOC99]	y1(GLIBC_2.0) [SUSv3]
y1f(GLIBC_2.0) [ISOC99]	y1l(GLIBC_2.0) [ISOC99]	yn(GLIBC_2.0) [SUSv3]	ynf(GLIBC_2.0) [ISOC99]
ynl(GLIBC_2.0) [ISOC99]			

1907

1908

1909

1910

1911

1912

1913

1914

1915

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

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

~~[3]. this specification~~

~~[4]. SUSv2~~

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

1916

Table 11-26 libm - Math Data Interfaces

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

1917

1918

1919

~~Referenced Specification(s)~~

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

1920

signgam(GLIBC_2.0) [SUSv3]			
----------------------------	--	--	--

11.5 Data Definitions for libm

1921

1922

1923

1924

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.

1925

1926

1927

1928

~~These definitions are intended to supplement those provided in~~ Where an interface is defined as requiring a particular system header file all of the ~~referenced underlying~~ data definitions for that system header file presented here shall be in effect.

1929

1930

1931

1932

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 supplement - not to replace - source interface definition specifications.

1933

1934

1935

1936

This specification uses ~~ISO/IEC 9899~~ the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

11.5.1 complex.h

```

1937
1938     extern double cabs(double complex);
1939     extern float cabsf(float complex);
1940     extern long double cabsl(long double complex);
1941     extern double complex cacos(double complex);
1942     extern float complex cacosf(float complex);
1943     extern double complex cacosh(double complex);
1944     extern float complex cacoshf(float complex);
1945     extern long double complex cacoshl(long double complex);
1946     extern long double complex cacosl(long double complex);
1947     extern double carg(double complex);
1948     extern float cargf(float complex);
1949     extern long double cargl(long double complex);
1950     extern double complex casin(double complex);
1951     extern float complex casinf(float complex);
1952     extern double complex casinh(double complex);
1953     extern float complex casinhf(float complex);
1954     extern long double complex casinhl(long double complex);
1955     extern long double complex casinl(long double complex);
1956     extern double complex catan(double complex);
1957     extern float complex catanf(float complex);
1958     extern double complex catanh(double complex);
1959     extern float complex catanhf(float complex);
1960     extern long double complex catanhl(long double complex);
1961     extern long double complex catanl(long double complex);
1962     extern double complex ccos(double complex);
1963     extern float complex ccosf(float complex);
1964     extern double complex ccosh(double complex);
1965     extern float complex ccoshf(float complex);
1966     extern long double complex ccoshl(long double complex);
1967     extern long double complex ccosl(long double complex);
1968     extern double complex cexp(double complex);
1969     extern float complex cexpf(float complex);
1970     extern long double complex cexpl(long double complex);
1971     extern double cimag(double complex);
1972     extern float cimagf(float complex);
1973     extern long double cimagl(long double complex);
1974     extern double complex clog(double complex);
1975     extern float complex clog10f(float complex);
1976     extern long double complex clog10l(long double complex);
1977     extern float complex clogf(float complex);
1978     extern long double complex clogl(long double complex);
1979     extern double complex conj(double complex);
1980     extern float complex conjf(float complex);
1981     extern long double complex conjl(long double complex);
1982     extern double complex cpow(double complex, double complex);
1983     extern float complex cpowf(float complex, float complex);
1984     extern long double complex cpowl(long double complex, long double
1985     complex);
1986     extern double complex cproj(double complex);
1987     extern float complex cprojf(float complex);
1988     extern long double complex cprojl(long double complex);
1989     extern double creal(double complex);
1990     extern float crealf(float complex);
1991     extern long double creall(long double complex);
1992     extern double complex csin(double complex);
1993     extern float complex csinf(float complex);
1994     extern double complex csinh(double complex);
1995     extern float complex csinhf(float complex);
1996     extern long double complex csinhl(long double complex);
1997     extern long double complex csinl(long double complex);
1998     extern double complex csqrt(double complex);

```

```

1999 extern float complex csqrtf(float complex);
2000 extern long double complex csqrtl(long double complex);
2001 extern double complex ctan(double complex);
2002 extern float complex ctanf(float complex);
2003 extern double complex ctanh(double complex);
2004 extern float complex ctanhf(float complex);
2005 extern long double complex ctanhl(long double complex);
2006 extern long double complex ctanl(long double complex);

```

11.5.2 fenv.h

```

2007
2008 #define FE_INEXACT      0x08
2009 #define FE_UNDERFLOW   0x10
2010 #define FE_OVERFLOW    0x20
2011 #define FE_DIVBYZERO   0x40
2012 #define FE_INVALID     0x80
2013
2014 #define FE_ALL_EXCEPT \
2015     (FE_INEXACT | FE_DIVBYZERO | FE_UNDERFLOW | FE_OVERFLOW |
2016     FE_INVALID)
2017
2018 #define FE_TONEAREST    0
2019 #define FE_TOWARDZERO  0x1
2020 #define FE_UPWARD      0x2
2021 #define FE_DOWNWARD    0x3
2022
2023 typedef unsigned int fexcept_t;
2024
2025 typedef struct {
2026     fexcept_t fpc;
2027     void *ieee_instruction_pointer;
2028 }
2029     fenv_t;
2030
2031 #define FE_DFL_ENV      ((__const fenv_t *) -1)
2032
2033 extern int feclearexcept(int);
2034 extern int fegetenv(fenv_t *);
2035 extern int fegetexceptflag(fexcept_t *, int);
2036 extern int fegetround(void);
2037 extern int feholdexcept(fenv_t *);
2038 extern int feraiseexcept(int);
2039 extern int fesetenv(const fenv_t *);
2040 extern int fesetexceptflag(const fexcept_t *, int);
2041 extern int fesetround(int);
2042 extern int fetestexcept(int);
2043 extern int feupdateenv(const fenv_t *);
2044

```

11.5.23 math.h

```

2045
2046 #define fpclassify(x) \
2047     (sizeof (x) == sizeof (float) ? __fpclassifyf (x) : __fpclassify
2048     (x) )
2049 #define signbit(x) \
2050     (sizeof (x) == sizeof (float)? __signbitf (x): __signbit (x))
2051
2052 #define FP_ILOGB0      -2147483647
2053 #define FP_ILOGBNAN   2147483647
2054
2055 extern int __finite(double);
2056 extern int __finitef(float);

```

```

2057     extern int __finitel(long double);
2058     extern int __isinf(double);
2059     extern int __isinff(float);
2060     extern int __isinfl(long double);
2061     extern int __isnan(double);
2062     extern int __isnanf(float);
2063     extern int __isnanl(long double);
2064     extern int __signbit(double);
2065     extern int __signbitf(float);
2066     extern int __fpclassify(double);
2067     extern int __fpclassifyf(float);
2068     extern int __fpclassifyl(long double);
2069     extern int signgam(void);
2070     extern double copysign(double, double);
2071     extern int finite(double);
2072     extern double frexp(double, int *);
2073     extern double ldexp(double, int);
2074     extern double modf(double, double *);
2075     extern double acos(double);
2076     extern double acosh(double);
2077     extern double asinh(double);
2078     extern double atanh(double);
2079     extern double asin(double);
2080     extern double atan(double);
2081     extern double atan2(double, double);
2082     extern double cbrt(double);
2083     extern double ceil(double);
2084     extern double cos(double);
2085     extern double cosh(double);
2086     extern double erf(double);
2087     extern double erfc(double);
2088     extern double exp(double);
2089     extern double expm1(double);
2090     extern double fabs(double);
2091     extern double floor(double);
2092     extern double fmod(double, double);
2093     extern double gamma(double);
2094     extern double hypot(double, double);
2095     extern int ilogb(double);
2096     extern double j0(double);
2097     extern double j1(double);
2098     extern double jn(int, double);
2099     extern double lgamma(double);
2100     extern double log(double);
2101     extern double log10(double);
2102     extern double loglp(double);
2103     extern double logb(double);
2104     extern double nextafter(double, double);
2105     extern double pow(double, double);
2106     extern double remainder(double, double);
2107     extern double rint(double);
2108     extern double scalb(double, double);
2109     extern double sin(double);
2110     extern double sinh(double);
2111     extern double sqrt(double);
2112     extern double tan(double);
2113     extern double tanh(double);
2114     extern double y0(double);
2115     extern double y1(double);
2116     extern double yn(int, double);
2117     extern float copysignf(float, float);
2118     extern long double copysignl(long double, long double);
2119     extern int finitef(float);
2120     extern int finitel(long double);

```

```

2121 extern float frexpf(float, int *);
2122 extern long double frexpl(long double, int *);
2123 extern float ldexpf(float, int);
2124 extern long double ldexpl(long double, int);
2125 extern float modff(float, float *);
2126 extern long double modfl(long double, long double *);
2127 extern double scalbln(double, long int);
2128 extern float scalblnf(float, long int);
2129 extern long double scalblnl(long double, long int);
2130 extern double scalbn(double, int);
2131 extern float scalbnf(float, int);
2132 extern long double scalbnl(long double, int);
2133 extern float acosf(float);
2134 extern float acoshf(float);
2135 extern long double acoshl(long double);
2136 extern long double acosl(long double);
2137 extern float asinf(float);
2138 extern float asinhf(float);
2139 extern long double asinhl(long double);
2140 extern long double asinl(long double);
2141 extern float atan2f(float, float);
2142 extern long double atan2l(long double, long double);
2143 extern float atanf(float);
2144 extern float atanhf(float);
2145 extern long double atanhhl(long double);
2146 extern long double atanl(long double);
2147 extern float cbrtf(float);
2148 extern long double cbrtl(long double);
2149 extern float ceilf(float);
2150 extern long double ceill(long double);
2151 extern float cosf(float);
2152 extern float coshf(float);
2153 extern long double coshl(long double);
2154 extern long double cosl(long double);
2155 extern float dremf(float, float);
2156 extern long double dreml(long double, long double);
2157 extern float erfcf(float);
2158 extern long double erfcl(long double);
2159 extern float erff(float);
2160 extern long double erfl(long double);
2161 extern double exp2(double);
2162 extern float exp2f(float);
2163 extern long double exp2l(long double);
2164 extern float expf(float);
2165 extern long double expl(long double);
2166 extern float expmlf(float);
2167 extern long double expmll(long double);
2168 extern float fabsf(float);
2169 extern long double fabsl(long double);
2170 extern double fdim(double, double);
2171 extern float fdimf(float, float);
2172 extern long double fdiml(long double, long double);
2173 extern float floorf(float);
2174 extern long double floorl(long double);
2175 extern double fma(double, double, double);
2176 extern float fmaf(float, float, float);
2177 extern long double fmal(long double, long double, long double);
2178 extern double fmax(double, double);
2179 extern float fmaxf(float, float);
2180 extern long double fmaxl(long double, long double);
2181 extern double fmin(double, double);
2182 extern float fminf(float, float);
2183 extern long double fminl(long double, long double);
2184 extern float fmodf(float, float);

```

```

2185     extern long double fmodl(long double, long double);
2186     extern float gammaf(float);
2187     extern long double gammal(long double);
2188     extern float hypotf(float, float);
2189     extern long double hypotl(long double, long double);
2190     extern int ilogbf(float);
2191     extern int ilogbl(long double);
2192     extern float j0f(float);
2193     extern long double j0l(long double);
2194     extern float j1f(float);
2195     extern long double j1l(long double);
2196     extern float jnf(int, float);
2197     extern long double jnl(int, long double);
2198     extern double lgamma_r(double, int *);
2199     extern float lgammaf(float);
2200     extern float lgammaf_r(float, int *);
2201     extern long double lgammal(long double);
2202     extern long double lgammal_r(long double, int *);
2203     extern long long int llrint(double);
2204     extern long long int llrintf(float);
2205     extern long long int llrintl(long double);
2206     extern long long int llround(double);
2207     extern long long int llroundf(float);
2208     extern long long int llroundl(long double);
2209     extern float log10f(float);
2210     extern long double log10l(long double);
2211     extern float log1pf(float);
2212     extern long double log1pl(long double);
2213     extern double log2(double);
2214     extern float log2f(float);
2215     extern long double log2l(long double);
2216     extern float logbf(float);
2217     extern long double logbl(long double);
2218     extern float logf(float);
2219     extern long double logl(long double);
2220     extern long int lrint(double);
2221     extern long int lrintf(float);
2222     extern long int lrintl(long double);
2223     extern long int lround(double);
2224     extern long int lroundf(float);
2225     extern long int lroundl(long double);
2226     extern int matherr(struct exception *);
2227     extern double nan(const char *);
2228     extern float nanf(const char *);
2229     extern long double nanl(const char *);
2230     extern double nearbyint(double);
2231     extern float nearbyintf(float);
2232     extern long double nearbyintl(long double);
2233     extern float nextafterf(float, float);
2234     extern long double nextafterl(long double, long double);
2235     extern double nexttoward(double, long double);
2236     extern float nexttowardf(float, long double);
2237     extern long double nexttowardl(long double, long double);
2238     extern double pow10(double);
2239     extern float pow10f(float);
2240     extern long double pow10l(long double);
2241     extern float powf(float, float);
2242     extern long double powl(long double, long double);
2243     extern float remainderf(float, float);
2244     extern long double remainderl(long double, long double);
2245     extern double remquo(double, double, int *);
2246     extern float remquof(float, float, int *);
2247     extern long double remquol(long double, long double, int *);
2248     extern float rintf(float);

```

```

2249 extern long double rintl(long double);
2250 extern double round(double);
2251 extern float roundf(float);
2252 extern long double roundl(long double);
2253 extern float scalbf(float, float);
2254 extern long double scalbl(long double, long double);
2255 extern double significand(double);
2256 extern float significandf(float);
2257 extern long double significandl(long double);
2258 extern void sincos(double, double *, double *);
2259 extern void sincosf(float, float *, float *);
2260 extern void sincosl(long double, long double *, long double *);
2261 extern float sinf(float);
2262 extern float sinhf(float);
2263 extern long double sinhl(long double);
2264 extern long double sinl(long double);
2265 extern float sqrtf(float);
2266 extern long double sqrtl(long double);
2267 extern float tanf(float);
2268 extern float tanhf(float);
2269 extern long double tanhl(long double);
2270 extern long double tanl(long double);
2271 extern double tgamma(double);
2272 extern float tgammaf(float);
2273 extern long double tgamma1(long double);
2274 extern double trunc(double);
2275 extern float truncf(float);
2276 extern long double trunc1(long double);
2277 extern float y0f(float);
2278 extern long double y0l(long double);
2279 extern float y1f(float);
2280 extern long double y1l(long double);
2281 extern float ynf(int, float);
2282 extern long double ynl(int, long double);
2283 extern int __fpclassifyl(long double);
2284 extern int __fpclassifyl(long double);
2285 extern int __signbitl(long double);
2286 extern int __signbitl(long double);
2287 extern int __signbitl(long double);
2288 extern long double exp2l(long double);
2289 extern long double exp2l(long double);

```

11.6 Interfaces for libpthread

2290 Table 11-27 defines the library name and shared object name for the libpthread
 2291 library

2292 **Table 11-27 libpthread Definition**

2293 Library:	libpthread
SONAME:	libpthread.so.0

2294 The behavior of the interfaces in this library is specified by the following specifica-
 2295 tions:

2296 [LFS] Large File Support
 [LSB] ~~this specification~~ This Specification
 [SUSv3] ISO POSIX (2003)

11.6.1 Realtime Threads

11.6.1.1 Interfaces for Realtime Threads

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

Table 11-28 libpthread - Realtime Threads Function Interfaces

<code>pthread_attr_getinheritsched(GLIBC_2.0)</code> [1]	<code>pthread_attr_getscope(GLIBC_2.0)</code> [1]	<code>pthread_attr_setschedpolicy(GLIBC_2.0)</code> [1]	<code>pthread_getschedparam(GLIBC_2.0)</code> [1]	
<code>pthread_attr_setschedpolicy(GLIBC_2.0)</code> [1]	<code>pthread_attr_setinheritsched(GLIBC_2.0)</code> [1]	<code>pthread_attr_setscope(GLIBC_2.0)</code> [1]	<code>pthread_setschedparam(GLIBC_2.0)</code> [1]	

Referenced Specification(s)

[1]. ISO POSIX (2003)

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

11.6.2 Advanced Realtime Threads

11.6.2.1 Interfaces for Advanced Realtime Threads

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

11.6.3 Posix Threads

11.6.3.1 Interfaces for Posix Threads

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

Table 11-29 libpthread - Posix Threads Function Interfaces

<code>pthread_cleanup_pop(GLIBC_2.0)</code> [1]	<code>pthread_cond_broadcast(GLIBC_2.3.2)</code> [2]	<code>pthread_join(GLIBC_2.0)</code> [2]	<code>pthread_rwlock_destroy(GLIBC_2.1)</code> [2]	<code>pthread_setcancelstate(GLIBC_2.1)</code> [2]
<code>pthread_cleanup_push(GLIBC_2.0)</code> [1]	<code>pthread_cond_destroy(GLIBC_2.3.2)</code> [2]	<code>pthread_key_create(GLIBC_2.0)</code> [2]	<code>pthread_rwlock_init(GLIBC_2.1)</code> [2]	<code>pthread_setspecific(GLIBC_2.0)</code> [2]

<code>pthread_attr_destroy</code> (GLIBC_2.0) [2]	<code>pthread_cond_init</code> (GLIBC_2.3.2) [2]	<code>pthread_key_delete</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_rdlock</code> (GLIBC_2.1) [2]	<code>pthread_sigmask</code> (GLIBC_2.0) [2]
<code>pthread_attr_getdetachstate</code> (GLIBC_2.0) [2]	<code>pthread_cond_signal</code> (GLIBC_2.3.2) [2]	<code>pthread_kill</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_timedrdlock</code> (GLIBC_2.2) [2]	<code>pthread_testcancel</code> (GLIBC_2.0) [2]
<code>pthread_attr_getguardsize</code> (GLIBC_2.1) [2]	<code>pthread_cond_timedwait</code> (GLIBC_2.3.2) [2]	<code>pthread_mutex_destroy</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_timedwrlock</code> (GLIBC_2.2) [2]	<code>sem_close</code> (GLIBC_2.1.1) [2]
<code>pthread_attr_getschedparam</code> (GLIBC_2.0) [2]	<code>pthread_cond_wait</code> (GLIBC_2.3.2) [2]	<code>pthread_mutex_init</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_tryrdlock</code> (GLIBC_2.1) [2]	<code>sem_destroy</code> (GLIBC_2.1) [2]
<code>pthread_attr_getstack</code> (GLIBC_2.2) [2]	<code>pthread_cond_attr_destroy</code> (GLIBC_2.0) [2]	<code>pthread_mutex_lock</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_trywrlock</code> (GLIBC_2.1) [2]	<code>sem_getvalue</code> (GLIBC_2.1) [2]
<code>pthread_attr_getstackaddr</code> (GLIBC_2.1) [2]	<code>pthread_cond_attr_getpshared</code> (GLIBC_2.2) [2]	<code>pthread_mutex_trylock</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_unlock</code> (GLIBC_2.1) [2]	<code>sem_init</code> (GLIBC_2.1) [2]
<code>pthread_attr_getstacksize</code> (GLIBC_2.1) [2]	<code>pthread_cond_attr_init</code> (GLIBC_2.0) [2]	<code>pthread_mutex_unlock</code> (GLIBC_2.0) [2]	<code>pthread_rwlock_wrlock</code> (GLIBC_2.1) [2]	<code>sem_open</code> (GLIBC_2.1.1) [2]
<code>pthread_attr_init</code> (GLIBC_2.1) [2]	<code>pthread_cond_attr_setpshared</code> (GLIBC_2.2) [2]	<code>pthread_mutexattr_destroy</code> (GLIBC_2.0) [2]	<code>pthread_rwlockattr_destroy</code> (GLIBC_2.1) [2]	<code>sem_post</code> (GLIBC_2.1) [2]
<code>pthread_attr_setdetachstate</code> (GLIBC_2.0) [2]	<code>pthread_create</code> (GLIBC_2.1) [2]	<code>pthread_mutexattr_getpshared</code> (GLIBC_2.2) [2]	<code>pthread_rwlockattr_getpshared</code> (GLIBC_2.1) [2]	<code>sem_timedwait</code> (GLIBC_2.2) [2]
<code>pthread_attr_setguardsize</code> (GLIBC_2.1) [2]	<code>pthread_detach</code> (GLIBC_2.0) [2]	<code>pthread_mutexattr_gettype</code> (GLIBC_2.1) [2]	<code>pthread_rwlockattr_init</code> (GLIBC_2.1) [2]	<code>sem_trywait</code> (GLIBC_2.1) [2]
<code>pthread_attr_setschedparam</code> (GLIBC_2.0) [2]	<code>pthread_equality</code> (GLIBC_2.0) [2]	<code>pthread_mutexattr_init</code> (GLIBC_2.0) [2]	<code>pthread_rwlockattr_setpshared</code> (GLIBC_2.1) [2]	<code>sem_unlink</code> (GLIBC_2.1.1) [2]
<code>pthread_attr_setstackaddr</code> (GLIBC_2.1) [2]	<code>pthread_exit</code> (GLIBC_2.0) [2]	<code>pthread_mutexattr_setpshared</code> (GLIBC_2.2) [2]	<code>pthread_self</code> (GLIBC_2.0) [2]	<code>sem_wait</code> (GLIBC_2.1) [2]

[2]) [2]		
pthread_attr_setstacksize(GLIBC_2.1) [2]	pthread_getconcurrency(GLIBC_2.1) [2]	pthread_mutexattr_settype(GLIBC_2.1) [2]	pthread_setcancelstate(GLIBC_2.0) [2]	
pthread_cancel(GLIBC_2.0) [2]	pthread_getspecific(GLIBC_2.0) [2]	pthread_once(GLIBC_2.0) [2]	pthread_setcanceltype(GLIBC_2.0) [2]	

2314

2315

Referenced Specification(s)

2316

[1]. this specification

2317

[2]. ISO POSIX (2003)

_pthread_cleanup_pop(GLIBC_2.0) [LSB]	_pthread_cleanup_push(GLIBC_2.0) [LSB]	pthread_attr_destroy(GLIBC_2.0) [SUSv3]	pthread_attr_getdetachstate(GLIBC_2.0) [SUSv3]
pthread_attr_getguardsize(GLIBC_2.1) [SUSv3]	pthread_attr_getschedparam(GLIBC_2.0) [SUSv3]	pthread_attr_getstack(GLIBC_2.2) [SUSv3]	pthread_attr_getstackaddr(GLIBC_2.1) [SUSv3]
pthread_attr_getstacksize(GLIBC_2.1) [SUSv3]	pthread_attr_init(GLIBC_2.1) [SUSv3]	pthread_attr_setdetachstate(GLIBC_2.0) [SUSv3]	pthread_attr_setguardsize(GLIBC_2.1) [SUSv3]
pthread_attr_setschedparam(GLIBC_2.0) [SUSv3]	pthread_attr_setstackaddr(GLIBC_2.1) [SUSv3]	pthread_attr_setstacksize(GLIBC_2.1) [SUSv3]	pthread_cancel(GLIBC_2.0) [SUSv3]
pthread_cond_broadcast(GLIBC_2.3.2) [SUSv3]	pthread_cond_destroy(GLIBC_2.3.2) [SUSv3]	pthread_cond_init(GLIBC_2.3.2) [SUSv3]	pthread_cond_signal(GLIBC_2.3.2) [SUSv3]
pthread_cond_timedwait(GLIBC_2.3.2) [SUSv3]	pthread_cond_wait(GLIBC_2.3.2) [SUSv3]	pthread_condattr_destroy(GLIBC_2.0) [SUSv3]	pthread_condattr_getpshared(GLIBC_2.2) [SUSv3]
pthread_condattr_init(GLIBC_2.0) [SUSv3]	pthread_condattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_create(GLIBC_2.1) [SUSv3]	pthread_detach(GLIBC_2.0) [SUSv3]
pthread_equal(GLIBC_2.0) [SUSv3]	pthread_exit(GLIBC_2.0) [SUSv3]	pthread_getconcurrency(GLIBC_2.1) [SUSv3]	pthread_getspecific(GLIBC_2.0) [SUSv3]
pthread_join(GLIBC_2.0) [SUSv3]	pthread_key_create(GLIBC_2.0) [SUSv3]	pthread_key_delete(GLIBC_2.0) [SUSv3]	pthread_kill(GLIBC_2.0) [SUSv3]
pthread_mutex_destroy(GLIBC_2.0) [SUSv3]	pthread_mutex_init(GLIBC_2.0) [SUSv3]	pthread_mutex_lock(GLIBC_2.0) [SUSv3]	pthread_mutex_trylock(GLIBC_2.0) [SUSv3]
pthread_mutex_unlock(GLIBC_2.0)	pthread_mutexattr_destroy(GLIBC_	pthread_mutexattr_getpshared(GLI	pthread_mutexattr_gettype(GLIBC_

[SUSv3]	2.0) [SUSv3]	BC_2.2) [SUSv3]	2.1) [SUSv3]
pthread_mutexattr_init(GLIBC_2.0) [SUSv3]	pthread_mutexattr_setshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_settype(GLIBC_2.1) [SUSv3]	pthread_once(GLIBC_2.0) [SUSv3]
pthread_rwlock_destroy(GLIBC_2.1) [SUSv3]	pthread_rwlock_init(GLIBC_2.1) [SUSv3]	pthread_rwlock_rdlock(GLIBC_2.1) [SUSv3]	pthread_rwlock_timedrdlock(GLIBC_2.2) [SUSv3]
pthread_rwlock_timedwrlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_tryrdlock(GLIBC_2.1) [SUSv3]	pthread_rwlock_trywrlock(GLIBC_2.1) [SUSv3]	pthread_rwlock_unlock(GLIBC_2.1) [SUSv3]
pthread_rwlock_wrlock(GLIBC_2.1) [SUSv3]	pthread_rwlockat_destroy(GLIBC_2.1) [SUSv3]	pthread_rwlockat_getpshared(GLIBC_2.1) [SUSv3]	pthread_rwlockat_init(GLIBC_2.1) [SUSv3]
pthread_rwlockat_setpshared(GLIBC_2.1) [SUSv3]	pthread_self(GLIBC_2.0) [SUSv3]	pthread_setcancelstate(GLIBC_2.0) [SUSv3]	pthread_setcanceltype(GLIBC_2.0) [SUSv3]
pthread_setconcurrency(GLIBC_2.1) [SUSv3]	pthread_setspecific(GLIBC_2.0) [SUSv3]	pthread_sigmask(GLIBC_2.0) [SUSv3]	pthread_testcancel(GLIBC_2.0) [SUSv3]
sem_close(GLIBC_2.1.1) [SUSv3]	sem_destroy(GLIBC_2.1) [SUSv3]	sem_getvalue(GLIBC_2.1) [SUSv3]	sem_init(GLIBC_2.1) [SUSv3]
sem_open(GLIBC_2.1.1) [SUSv3]	sem_post(GLIBC_2.1) [SUSv3]	sem_timedwait(GLIBC_2.2) [SUSv3]	sem_trywait(GLIBC_2.1) [SUSv3]
sem_unlink(GLIBC_2.1.1) [SUSv3]	sem_wait(GLIBC_2.1) [SUSv3]		

2318

11.6.4 Thread aware versions of libc interfaces

2319

11.6.4.1 Interfaces for Thread aware versions of libc interfaces

2320

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.

2321

2322

2323

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

2324

lseek64(GLIBC_2.2) [1]	pread(GLIBC_2.2) [2]	pwrite(GLIBC_2.2) [2]		
open64(GLIBC_2.2) [1]	pread64(GLIBC_2.2) [1]	pwrite64(GLIBC_2.2) [1]		

2325

Referenced Specification(s)

2326

[1]:

2327

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

) [SUSv3]	2.2) [LFS]		
-----------	------------	--	--

2328

11.7 Data Definitions for libpthread

2329

2330

2331

2332

2333

2334

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.

2335

2336

2337

2338

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.

2339

This specification uses the ~~Large File Support~~ ISO C (1999)

2340

2341

2342

2343

~~[2]~~ C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

11.7.1 pthread.h

2344

2345

2346

2347

2348

2349

2350

2351

2352

2353

2354

2355

2356

2357

2358

2359

2360

2361

2362

2363

2364

2365

2366

2367

2368

2369

2370

2371

2372

2373

2374

2375

2376

2377

2378

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

```

2379         struct sched_param
2380     __schedparam;
2381         int __inheritsched;
2382         int __scope;
2383         size_t __guardsize;
2384         int __stackaddr_set;
2385         void *__stackaddr;
2386         unsigned long int __stacksize;}
2387     pthread_attr_t *, struct
2388     sched_param {
2389         int sched_priority;}
2390
2391     *);
2392     extern int pthread_attr_getschedpolicy(const typedef struct {
2393         int __detachstate;
2394         int __schedpolicy;
2395         struct sched_param
2396     __schedparam;
2397         int __inheritsched;
2398         int __scope;
2399         size_t __guardsize;
2400         int __stackaddr_set;
2401         void *__stackaddr;
2402         unsigned long int __stacksize;}
2403     pthread_attr_t *, int *);
2404     extern int pthread_attr_getscope(const typedef struct {
2405         int __detachstate;
2406         int __schedpolicy;
2407         struct sched_param __schedparam;
2408         int __inheritsched;
2409         int __scope;
2410         size_t __guardsize;
2411         int __stackaddr_set;
2412         void *__stackaddr;
2413         unsigned long int __stacksize;}
2414     pthread_attr_t *, int *);
2415     extern int pthread_attr_init(pthread_attr_t *);
2416     extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
2417     extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
2418     extern int pthread_attr_setschedparam(pthread_attr_t *, const struct
2419     sched_param {
2420         int sched_priority;}
2421
2422     *);
2423     extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
2424     extern int pthread_attr_setscope(pthread_attr_t *, int);
2425     extern int pthread_cancel(typedef unsigned long int pthread_t);
2426     extern int pthread_cond_broadcast(pthread_cond_t *);
2427     extern int pthread_cond_destroy(pthread_cond_t *);
2428     extern int pthread_cond_init(pthread_cond_t *, const typedef struct {
2429         int __dummy;}
2430
2431         pthread_condattr_t *);
2432     extern int pthread_cond_signal(pthread_cond_t *);
2433     extern int pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
2434     const struct timespec {
2435         time_t tv_sec; long int tv_nsec;}
2436
2437     *);
2438     extern int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
2439     extern int pthread_condattr_destroy(pthread_condattr_t *);
2440     extern int pthread_condattr_init(pthread_condattr_t *);
2441     extern int pthread_create(pthread_t *, const typedef struct {
2442         int __detachstate;

```

```

2443         int __schedpolicy;
2444         struct sched_param __schedparam;
2445         int __inheritsched;
2446         int __scope;
2447         size_t __guardsize;
2448         int __stackaddr_set;
2449         void *__stackaddr;
2450         unsigned long int __stacksize;
2451         pthread_attr_t *,
2452         void *(*__start_routine) (void *p1)
2453         , void *);
2454 extern int pthread_detach(typedef unsigned long int pthread_t);
2455 extern int pthread_equal(typedef unsigned long int pthread_t,
2456                         typedef unsigned long int pthread_t);
2457 extern void pthread_exit(void *);
2458 extern int pthread_getschedparam(typedef unsigned long int pthread_t,
2459                                 int *, struct sched_param {
2460                                 int sched_priority;
2461
2462                                 });
2463 extern void *pthread_getspecific(typedef unsigned int pthread_key_t);
2464 extern int pthread_join(typedef unsigned long int pthread_t, void **);
2465 extern int pthread_key_create(pthread_key_t *, void (*destr_func) (void
2466 *)
2467 );
2468 extern int pthread_key_delete(typedef unsigned int pthread_key_t);
2469 extern int pthread_mutex_destroy(pthread_mutex_t *);
2470 extern int pthread_mutex_init(pthread_mutex_t *, const typedef struct
2471 {
2472         int __mutexkind;
2473
2474         pthread_mutexattr_t *);
2475 extern int pthread_mutex_lock(pthread_mutex_t *);
2476 extern int pthread_mutex_trylock(pthread_mutex_t *);
2477 extern int pthread_mutex_unlock(pthread_mutex_t *);
2478 extern int pthread_mutexattr_destroy(pthread_mutexattr_t *);
2479 extern int pthread_mutexattr_init(pthread_mutexattr_t *);
2480 extern int pthread_once(pthread_once_t *, void (*init_routine) (void)
2481 );
2482 extern int pthread_rwlock_destroy(pthread_rwlock_t *);
2483 extern int pthread_rwlock_init(pthread_rwlock_t *,
2484 pthread_rwlockattr_t *);
2485 extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
2486 extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *);
2487 extern int pthread_rwlock_trywrlock(pthread_rwlock_t *);
2488 extern int pthread_rwlock_unlock(pthread_rwlock_t *);
2489 extern int pthread_rwlock_wrlock(pthread_rwlock_t *);
2490 extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
2491 extern int pthread_rwlockattr_getpshared(const typedef struct {
2492         int __lockkind; int
2493         __pshared; }
2494         pthread_rwlockattr_t *, int
2495         *);
2496 extern int pthread_rwlockattr_init(pthread_rwlockattr_t *);
2497 extern int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
2498 extern typedef unsigned long int pthread_t pthread_self(void);
2499 extern int pthread_setcancelstate(int, int *);
2500 extern int pthread_setcanceltype(int, int *);
2501 extern int pthread_setschedparam(typedef unsigned long int pthread_t,
2502 int, const struct sched_param {
2503         int sched_priority;
2504
2505         *);
2506 extern int pthread_setspecific(typedef unsigned int pthread_key_t,

```

```

2507         const void *);
2508 extern void pthread_testcancel(void);
2509 extern int pthread_attr_getguardsize(const typedef struct {
2510     int __detachstate;
2511     int __schedpolicy;
2512     struct sched_param __schedparam;
2513     int __inheritsched;
2514     int __scope;
2515     size_t __guardsize;
2516     int __stackaddr_set;
2517     void *__stackaddr;
2518     unsigned long int __stacksize;}
2519     pthread_attr_t *, size_t *);
2520 extern int pthread_attr_setguardsize(pthread_attr_t *,
2521     typedef unsigned long int
2522     size_t);
2523 extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
2524 extern int pthread_attr_getstackaddr(const typedef struct {
2525     int __detachstate;
2526     int __schedpolicy;
2527     struct sched_param __schedparam;
2528     int __inheritsched;
2529     int __scope;
2530     size_t __guardsize;
2531     int __stackaddr_set;
2532     void *__stackaddr;
2533     unsigned long int __stacksize;}
2534     pthread_attr_t *, void **);
2535 extern int pthread_attr_setstacksize(pthread_attr_t *,
2536     typedef unsigned long int
2537     size_t);
2538 extern int pthread_attr_getstacksize(const typedef struct {
2539     int __detachstate;
2540     int __schedpolicy;
2541     struct sched_param __schedparam;
2542     int __inheritsched;
2543     int __scope;
2544     size_t __guardsize;
2545     int __stackaddr_set;
2546     void *__stackaddr;
2547     unsigned long int __stacksize;}
2548     pthread_attr_t *, size_t *);
2549 extern int pthread_mutexattr_gettype(const typedef struct {
2550     int __mutexkind;}
2551     pthread_mutexattr_t *, int *);
2552 extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
2553 extern int pthread_getconcurrency(void);
2554 extern int pthread_setconcurrency(int);
2555 extern int pthread_attr_getstack(const typedef struct {
2556     int __detachstate;
2557     int __schedpolicy;
2558     struct sched_param __schedparam;
2559     int __inheritsched;
2560     int __scope;
2561     size_t __guardsize;
2562     int __stackaddr_set;
2563     void *__stackaddr;
2564     unsigned long int __stacksize;}
2565     pthread_attr_t *, void **, size_t *);
2566 extern int pthread_attr_setstack(pthread_attr_t *, void *,
2567     typedef unsigned long int size_t);
2568 extern int pthread_condattr_getpshared(const typedef struct {
2569     int __dummy;}
2570     pthread_condattr_t *, int *);

```

```

2571 extern int pthread_condattr_setpshared(pthread_condattr_t *, int);
2572 extern int pthread_mutexattr_getpshared(const typedef struct {
2573     int __mutexkind;}
2574     pthread_mutexattr_t *, int *);
2575 extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
2576 extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *, const struct
2577 timespec {
2578     time_t tv_sec; long int
2579     tv_nsec; }
2580     *);
2581 extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *, const struct
2582 timespec {
2583     time_t tv_sec; long int
2584     tv_nsec; }
2585     *);
2586 extern int __register_atfork(void (*prepare) (void)
2587     , void (*parent) (void)
2588     , void (*child) (void)
2589     , void *);
2590 extern int pthread_setschedprio(typedef unsigned long int pthread_t,
2591 int);
2592

```

11.7.2 semaphore.h

```

2594 extern int sem_close(sem_t *);
2595 extern int sem_destroy(sem_t *);
2596 extern int sem_getvalue(sem_t *, int *);
2597 extern int sem_init(sem_t *, int, unsigned int);
2598 extern sem_t *sem_open(const char *, int, ...);
2599 extern int sem_post(sem_t *);
2600 extern int sem_trywait(sem_t *);
2601 extern int sem_unlink(const char *);
2602 extern int sem_wait(sem_t *);
2603 extern int sem_timedwait(sem_t *, const struct timespec *);
2604

```

11.8 Interfaces for libgcc_s

ISO POSIX (2003) Table 11-31

11.7 Interfaces for libgcc_s

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

Table 11-31 libgcc_s Definition

Library:	libgcc_s
SONAME:	libgcc_s.so.1

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

[LSB] ~~this specification~~ This Specification

11.78.1 Unwind Library

11.78.1.1 Interfaces for Unwind Library

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

Table 11-32 libgcc_s - Unwind Library Function Interfaces

<code>_Unwind_Backtrace(GCC_3.3)</code> [1]	<code>_Unwind_ForcedUnwind(GCC_3.0)</code> [1]	<code>_Unwind_GetIP(GCC_3.0)</code> [1]	<code>_Unwind_RaiseException(GCC_3.0)</code> [1]	<code>_Unwind_SetIP(GCC_3.0)</code> [1]
<code>_Unwind_DeleteException(GCC_3.0)</code> [1]	<code>_Unwind_GetCFA(GCC_3.3)</code> [1]	<code>_Unwind_GetLanguageSpecificData(GCC_3.0)</code> [1]	<code>_Unwind_Resume(GCC_3.0)</code> [1]	
<code>_Unwind_FindEnclosingFunction(GCC_3.3)</code> [1]	<code>_Unwind_GetDataRelBase(GCC_3.0)</code> [1]	<code>_Unwind_GetRegionStart(GCC_3.0)</code> [1]	<code>_Unwind_Resume_or_Rethrow(GCC_3.3)</code> [1]	
<code>_Unwind_FindFDE(GCC_3.0)</code> [1]	<code>_Unwind_GetGR(GCC_3.0)</code> [1]	<code>_Unwind_GetTextRelBase(GCC_3.0)</code> [1]	<code>_Unwind_SetGR(GCC_3.0)</code> [1]	

Referenced Specification(s)

[1]

<code>_Unwind_Backtrace(GCC_3.3)</code> [LSB]	<code>_Unwind_DeleteException(GCC_3.0)</code> [LSB]	<code>_Unwind_FindEnclosingFunction(GCC_3.3)</code> [LSB]	<code>_Unwind_FindFDE(GCC_3.0)</code> [LSB]
<code>_Unwind_ForcedUnwind(GCC_3.0)</code> [LSB]	<code>_Unwind_GetCFA(GCC_3.3)</code> [LSB]	<code>_Unwind_GetDataRelBase(GCC_3.0)</code> [LSB]	<code>_Unwind_GetGR(GCC_3.0)</code> [LSB]
<code>_Unwind_GetIP(GCC_3.0)</code> [LSB]	<code>_Unwind_GetLanguageSpecificData(GCC_3.0)</code> [LSB]	<code>_Unwind_GetRegionStart(GCC_3.0)</code> [LSB]	<code>_Unwind_GetTextRelBase(GCC_3.0)</code> [LSB]
<code>_Unwind_RaiseException(GCC_3.0)</code> [LSB]	<code>_Unwind_Resume(GCC_3.0)</code> [LSB]	<code>_Unwind_Resume_or_Rethrow(GCC_3.3)</code> [LSB]	<code>_Unwind_SetGR(GCC_3.0)</code> [LSB]
<code>_Unwind_SetIP(GCC_3.0)</code> [LSB]			

11.9 Data Definitions for libgcc_s

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

2625 interface is defined as requiring a particular system header file all of the data
2626 definitions for that system header file presented here shall be in effect.

2627 This section gives data definitions to promote binary application portability, not to
2628 repeat source interface definitions available elsewhere. System providers and
2629 application developers should use this ABI to supplement - not to replace - source
2630 interface definition specifications.

2631 This specification uses the ~~this specification~~ ISO C (1999)

11.8 Interface Definitions for `libgcc_s`

2632 The following interfaces are included in `libgcc_s` and are defined by this
2633 specification. Unless otherwise noted, these interfaces shall be included in the source
2634 standard.

2635 Other interfaces listed above for `libgcc_s` shall behave as described in the referenced
2636 base document.

11.9 Interfaces for `libdl`

2637 C Language as the reference programming language, and data definitions are
2638 specified in ISO C format. The C language is used here as a convenient notation.
2639 Using a C language description of these data objects does not preclude their use by
2640 other programming languages.

11.9.1 `unwind.h`

```
2641
2642 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2643 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2644 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2645 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2646                                         _Unwind_Stop_Fn, void *);
2647 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2648 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2649 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2650 _Unwind_Context
2651                                         *);
2652 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2653 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2654 _Unwind_Exception
2655                                         *);
2656 extern void _Unwind_Resume(struct _Unwind_Exception *);
2657 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2658 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2659 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2660 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2661 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2662                                         _Unwind_Stop_Fn, void *);
2663 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2664 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2665 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2666 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2667 _Unwind_Context
2668                                         *);
2669 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2670 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2671 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2672 _Unwind_Exception
2673                                         *);
2674 extern void _Unwind_Resume(struct _Unwind_Exception *);
```

```

2675 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2676 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2677 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2678                                         _Unwind_Stop_Fn, void *);
2679 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2680 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2681 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2682 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2683 _Unwind_Context
2684                                         *);
2685 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2686 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2687 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2688 _Unwind_Exception
2689                                         *);
2690 extern void _Unwind_Resume(struct _Unwind_Exception *);
2691 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2692 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2693 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2694 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2695 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2696                                         _Unwind_Stop_Fn, void *);
2697 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2698 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2699 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2700 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2701 _Unwind_Context
2702                                         *);
2703 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2704 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2705 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2706 _Unwind_Exception
2707                                         *);
2708 extern void _Unwind_Resume(struct _Unwind_Exception *);
2709 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2710 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2711 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2712 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2713 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2714                                         _Unwind_Stop_Fn, void *);
2715 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2716 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2717 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2718 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2719 _Unwind_Context
2720                                         *);
2721 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2722 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2723 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2724 _Unwind_Exception
2725                                         *);
2726 extern void _Unwind_Resume(struct _Unwind_Exception *);
2727 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2728 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2729 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2730 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2731 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2732                                         _Unwind_Stop_Fn, void *);
2733 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2734 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2735 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2736 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2737 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2738 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);

```

```

2739     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2740     _Unwind_Exception
2741     *);
2742     extern void _Unwind_Resume(struct _Unwind_Exception *);
2743     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2744     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2745     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2746     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2747     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2748     _Unwind_Stop_Fn, void *);
2749     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2750     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2751     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2752     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2753     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2754     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2755     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2756     _Unwind_Exception
2757     *);
2758     extern void _Unwind_Resume(struct _Unwind_Exception *);
2759     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2760     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2761     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2762     *);
2763     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2764     *);
2765     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2766     *);
2767     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2768     *);
2769     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2770     *);
2771     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2772     *);
2773     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2774     *);
2775     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2776     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2777     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2778     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2779     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2780     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2781     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2782     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2783
2784     _Unwind_Exception *);
2785     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2786
2787     _Unwind_Exception *);
2788     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2789
2790     _Unwind_Exception *);
2791     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2792
2793     _Unwind_Exception *);
2794     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2795
2796     _Unwind_Exception *);
2797     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2798
2799     _Unwind_Exception *);
2800     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2801
2802     _Unwind_Exception *);

```

```

2803     extern void *_Unwind_FindEnclosingFunction(void *);
2804     extern void *_Unwind_FindEnclosingFunction(void *);
2805     extern void *_Unwind_FindEnclosingFunction(void *);
2806     extern void *_Unwind_FindEnclosingFunction(void *);
2807     extern void *_Unwind_FindEnclosingFunction(void *);
2808     extern void *_Unwind_FindEnclosingFunction(void *);
2809     extern void *_Unwind_FindEnclosingFunction(void *);
2810     extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);

```

11.10 Interface Definitions for libgcc_s

2811 The interfaces defined on the following pages are included in libgcc_s and are
 2812 defined by this specification. Unless otherwise noted, these interfaces shall be
 2813 included in the source standard.

2814 Other interfaces listed in [Table 11-33](#) Section 11.8 shall behave as described in the
 2815 referenced base document.

_Unwind_DeleteException

Name

2816 `_Unwind_DeleteException` – private C++ error handling method

Synopsis

2817 `void _Unwind_DeleteException(struct _Unwind_Exception * object);`

Description

2818 `_Unwind_DeleteException()` deletes the given exception *object*. If a given
 2819 runtime resumes normal execution after catching a foreign exception, it will not
 2820 know how to delete that exception. Such an exception shall be deleted by calling
 2821 `_Unwind_DeleteException()`. This is a convenience function that calls the function
 2822 pointed to by the *exception_cleanup* field of the exception header.

_Unwind_Find_FDE

Name

2823 `_Unwind_Find_FDE` – private C++ error handling method

Synopsis

2824 `fde * _Unwind_Find_FDE(void * pc, struct dwarf_eh_bases * bases);`

Description

2825 `_Unwind_Find_FDE()` looks for the object containing *pc*, then inserts into *bases*.

_Unwind_ForcedUnwind

Name

2826 `_Unwind_ForcedUnwind` — private C++ error handling method

Synopsis

2827 `_Unwind_Reason_Code _Unwind_ForcedUnwind(struct _Unwind_Exception *`
2828 `object, _Unwind_Stop_Fn stop, void * stop_parameter);`

Description

2829 `_Unwind_ForcedUnwind()` raises an exception for forced unwinding, passing along
2830 the given exception *object*, which should have its *exception_class* and
2831 *exception_cleanup* fields set. The exception *object* has been allocated by the
2832 language-specific runtime, and has a language-specific format, except that it shall
2833 contain an `_Unwind_Exception` struct.

2834 Forced unwinding is a single-phase process. *stop* and *stop_parameter* control the
2835 termination of the unwind process instead of the usual personality routine query.
2836 *stop* is called for each unwind frame, with the parameteres described for the usual
2837 personality routine below, plus an additional *stop_parameter*.

Return Value

2838 When *stop* identifies the destination frame, it transfers control to the user code as
2839 appropriate without returning, normally after calling `_Unwind_DeleteException()`.
2840 If not, then it should return an `_Unwind_Reason_Code` value.

2841 If *stop* returns any reason code other than `_URC_NO_REASON`, then the stack state is
2842 indeterminate from the point of view of the caller of `_Unwind_ForcedUnwind()`.
2843 Rather than attempt to return, therefore, the unwind library should use the
2844 *exception_cleanup* entry in the exception, and then call `abort()`.

_URC_NO_REASON

2846 This is not the destination from. The unwind runtime will call frame's
2847 personality routine with the `_UA_FORCE_UNWIND` and `_UA_CLEANUP_PHASE` flag
2848 set in *actions*, and then unwind to the next frame and call the `stop()` function
2849 again.

_URC_END_OF_STACK

2851 In order to allow `_Unwind_ForcedUnwind()` to perform special processing
2852 when it reaches the end of the stack, the unwind runtime will call it after the last
2853 frame is rejected, with a `NULL` stack pointer in the context, and the `stop()`
2854 function shall catch this condition. It may return this code if it cannot handle
2855 end-of-stack.

_URC_FATAL_PHASE2_ERROR

2856 The `stop()` function may return this code for other fatal conditions like stack
2857 corruption.
2858

`_Unwind_GetDataRelBase`

Name

2859 `_Unwind_GetDataRelBase` – private IA64 C++ error handling method

Synopsis

2860 `_Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context * context);`

Description

2861 `_Unwind_GetDataRelBase()` returns the global pointer in register one for *context*.

`_Unwind_GetGR`

Name

2862 `_Unwind_GetGR` – private C++ error handling method

Synopsis

2863 `_Unwind_Word _Unwind_GetGR(struct _Unwind_Context * context, int index);`

Description

2864 `_Unwind_GetGR()` returns data at *index* found in *context*. The register is identified
2865 by its index: 0 to 31 are for the fixed registers, and 32 to 127 are for the stacked
2866 registers.

2867 During the two phases of unwinding, only GR1 has a guaranteed value, which is the
2868 global pointer of the frame referenced by the unwind *context*. If the register has its
2869 NAT bit set, the behavior is unspecified.

`_Unwind_GetIP`

Name

2870 `_Unwind_GetIP` – private C++ error handling method

Synopsis

2871 `_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);`

Description

2872 `_Unwind_GetIP()` returns the instruction pointer value for the routine identified by
2873 the unwind *context*.

_Unwind_GetLanguageSpecificData**Name**

2874 `_Unwind_GetLanguageSpecificData` – private C++ error handling method

Synopsis

2875 `_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *
2876 context, uint value);`

Description

2877 `_Unwind_GetLanguageSpecificData()` returns the address of the language specific
2878 data area for the current stack frame.

_Unwind_GetRegionStart**Name**

2879 `_Unwind_GetRegionStart` – private C++ error handling method

Synopsis

2880 `_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);`

Description

2881 `_Unwind_GetRegionStart()` routine returns the address (i.e., 0) of the beginning of
2882 the procedure or code fragment described by the current unwind descriptor block.

_Unwind_GetTextRelBase**Name**

2883 `_Unwind_GetTextRelBase` – private IA64 C++ error handling method

Synopsis

2884 `_Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context * context);`

Description

2885 `_Unwind_GetTextRelBase()` calls the abort method, then returns.

_Unwind_RaiseException

Name

2886 `_Unwind_RaiseException` – private C++ error handling method

Synopsis

2887 `_Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *`
2888 `object);`

Description

2889 `_Unwind_RaiseException()` raises an exception, passing along the given exception
2890 *object*, which should have its *exception_class* and *exception_cleanup* fields set.
2891 The exception object has been allocated by the language-specific runtime, and has a
2892 language-specific format, exception that it shall contain an `_Unwind_Exception`.

Return Value

2893 `_Unwind_RaiseException()` does not return unless an error condition is found. If
2894 an error condition occurs, an `_Unwind_Reason_Code` is returned:

`_URC_END_OF_STACK`

2895

2896 The unwinder encountered the end of the stack during phase one without
2897 finding a handler. The unwind runtime will not have modified the stack. The
2898 C++ runtime will normally call `uncaught_exception()` in this case.

`_URC_FATAL_PHASE1_ERROR`

2899

2900 The unwinder encountered an unexpected error during phase one, because of
2901 something like stack corruption. The unwind runtime will not have modified
2902 the stack. The C++ runtime will normally call `terminate()` in this case.

`_URC_FATAL_PHASE2_ERROR`

2903

2904 The unwinder encountered an unexpected error during phase two. This is
2905 usually a *throw*, which will call `terminate()`.

_Unwind_Resume

Name

2906 `_Unwind_Resume` – private C++ error handling method

Synopsis

2907 `void _Unwind_Resume(struct _Unwind_Exception * object);`

Description

2908 `_Unwind_Resume()` resumes propagation of an existing exception *object*. A call to
2909 this routine is inserted as the end of a landing pad that performs cleanup, but does
2910 not resume normal execution. It causes unwinding to proceed further.

_Unwind_SetGR

Name

2911 `_Unwind_SetGR` – private C++ error handling method

Synopsis

2912 `void _Unwind_SetGR(struct _Unwind_Context * context, int index, uint value);`

Description

2913 `_Unwind_SetGR()` sets the *value* of the register *indexed* for the routine identified by
2914 the unwind *context*.

_Unwind_SetIP

Name

2915 `_Unwind_SetIP` – private C++ error handling method

Synopsis

2916 `void _Unwind_SetIP(struct _Unwind_Context * context, uint value);`

Description

2917 `_Unwind_SetIP()` sets the *value* of the instruction pointer for the routine identified
2918 by the unwind *context*

11.11 Interfaces for libdl

2919 [Table 11-33](#) defines the library name and shared object name for the libdl library

2920 **Table 11-33 libdl Definition**

Library:	libdl
SONAME:	libdl.so.2

2922 The behavior of the interfaces in this library is specified by the following specifica-
2923 tions:

2924 ~~[LSB] this specification~~ This Specification
[SUSv3] ISO POSIX (2003)

11.9.11.1 Dynamic Loader

11.911.1.1 Interfaces for Dynamic Loader

2926 An LSB conforming implementation shall provide the architecture specific functions
2927 for Dynamic Loader specified in [Table 11-34](#), with the full mandatory functionality
2928 as described in the referenced underlying specification.

2929 **Table 11-34 libdl - Dynamic Loader Function Interfaces**

<code>dladdr</code> (GLIBC 2.0) [1]	<code>dlopen</code> (GLIBC 2.0) [2]	<code>dlsym</code> (GLIBC 2.0) [1]	<code>dlerror</code> (GLIBC 2.0) [2]	<code>dlclose</code> (GLIBC 2.1) [1]
----------------------------------------	----------------------------------------	---------------------------------------	-----------------------------------------	-----------------------------------------

2931

~~Referenced Specification(s)~~

2932

~~[1]~~

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

2933

11.12 Data Definitions for libdl

2934

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

2935

2936

2937

2938

2939

2940

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.

2941

2942

2943

2944

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

2945

~~[2]~~ C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

2946

2947

2948

11.12.1 dlfcn.h

2949

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

2950

2951

2952

2953

2954

11.13 Interfaces for libcrypt

2955

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

~~11.10 Interfaces for libcrypt~~

2956

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

2957

Table 11-35 libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

2958

2959

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

2960

2961

[SUSv3] ISO POSIX (2003)

11.4013.1 Encryption

11.4013.1.1 Interfaces for Encryption

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

Table 11-36 libcrypt - Encryption Function Interfaces

<code>crypt(GLIBC_2.0) [1]</code>	<code>encrypt(GLIBC_2.0) [1]</code>	<code>setkey(GLIBC_2.0) [1]</code>		
-----------------------------------	-------------------------------------	------------------------------------	--	--

Referenced Specification(s)

[1]. ISO POSIX (2003)

<code>crypt(GLIBC_2.0) [SUSv3]</code>	<code>encrypt(GLIBC_2.0) [SUSv3]</code>	<code>setkey(GLIBC_2.0) [SUSv3]</code>		
---------------------------------------	-----------------------------------------	----------------------------------------	--	--

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**

7 Library:	libz
SONAME:	libz.so.1

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

12.23.1 Curses

74 12.23.1.1 Interfaces for Curses

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

12.34 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 mwaddch(WINDOW *, int, int, const chtype);
205     extern int mwaddchnstr(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 scr1(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 vwscanw(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~~ This Specification

12.35.1 Utility Functions

12.35.1.1 Interfaces for Utility Functions

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

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

forkpty(GLIB C_2.0) [1]	login_tty(GLI BC_2.0) [1]	logwtmp(GLI BC_2.0) [1]		
login(GLIBC_ 2.0) [1]	logout(GLIBC _2.0) [1]	openpty(GLI BC_2.0) [1]		

407

408 *Referenced Specification(s)*

409 ~~[1], this specification~~

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

410

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-s390

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-s390`.

13.2 Package Architecture Considerations

8 All packages must specify an architecture of `s390`. A LSB runtime environment must
9 accept an architecture of `s390` even if the native architecture is different.

10 The `archnum` value in the Lead Section shall be `0x000E`.

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~~ This Specification [LSB]

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

_Unwind_Backtrace_Unwind_Backtrace [4]LSB]	_Unwind_GetDataRelBase[4]LSB]	_Unwind_RaiseException[4]LSB]
_Unwind_DeleteException[4]LSB]	_Unwind_GetGR_Unwind_GetGR [4]LSB]	_Unwind_Resume_Unwind_Resume [4]LSB]
_Unwind_FindEnclosingFunction[4]LSB]	_Unwind_GetIP_Unwind_GetIP [4]LSB]	_Unwind_Resume_or_Rethrow[4]LSB]
_Unwind_Find_FDE_Unwind_Find_FDE [4]LSB]	_Unwind_GetLanguageSpecificData[4]LSB]	_Unwind_SetGR_Unwind_SetGR [4]LSB]
_Unwind_ForcedUnwind_Unwind_ForcedUnwind [4]LSB]	_Unwind_GetRegionStart[4]LSB]	_Unwind_SetIP_Unwind_SetIP [4]LSB]
_Unwind_GetCFA_Unwind_GetCFA [4]LSB]	_Unwind_GetTextRelBase[4]LSB]	

4

Annex B GNU Free Documentation License (Informative)

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

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

B.1 PREAMBLE

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

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

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

B.2 APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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.