

Linux Standard Base Core Specification for S390 3.1

Linux Standard Base Core Specification for S390 3.1

Copyright © 2004, 2005 Free Standards Group

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

Portions of the text are copyrighted by the following parties:

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

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

Linux is a trademark of Linus Torvalds.

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

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

AMD is a trademark of Advanced Micro Devices, Inc.

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

PowerPC and PowerPC Architecture are trademarks of the IBM Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Contents

Foreword	vi
Introduction	vii
I Introductory Elements	8
1 Scope.....	9
1.1 General.....	9
1.2 Module Specific Scope.....	9
2 References	10
2.1 Normative References	10
2.2 Informative References/Bibliography	12
3 Requirements	14
3.1 Relevant Libraries	14
3.2 LSB Implementation Conformance	14
3.3 LSB Application Conformance.....	15
4 Definitions	17
5 Terminology	18
6 Documentation Conventions	20
II Executable and Linking Format (ELF)	21
7 Introduction.....	22
8 Low Level System Information.....	23
8.1 Machine Interface.....	23
8.2 Function Calling Sequence.....	24
8.3 Operating System Interface	24
8.4 Process Initialization.....	25
8.5 Coding Examples	25
8.6 Debug Information.....	25
9 Object Format	26
9.1 Introduction	26
9.2 ELF Header	26
9.3 Sections.....	26
9.4 Symbol Table	27
9.5 Relocation.....	27
10 Program Loading and Dynamic Linking	28
10.1 Introduction	28
10.2 Program Loading	28
10.3 Dynamic Linking.....	28
III Base Libraries	29
11 Libraries	30
11.1 Program Interpreter/Dynamic Linker	30
11.2 Interfaces for libc	30
11.3 Data Definitions for libc	44
11.4 Interfaces for libm	69
11.5 Data Definitions for libm.....	74
11.6 Interfaces for libpthread	80
11.7 Data Definitions for libpthread	82
11.8 Interfaces for libgcc_s	87
11.9 Data Definitions for libgcc_s.....	87
11.10 Interface Definitions for libgcc_s.....	91
11.11 Interfaces for libdl	96
11.12 Data Definitions for libdl	97

11.13 Interfaces for libcrypt.....	97
IV Utility Libraries.....	98
12 Libraries	99
12.1 Interfaces for libz.....	99
12.2 Data Definitions for libz	99
12.3 Interfaces for libncurses.....	100
12.4 Data Definitions for libncurses.....	100
12.5 Interfaces for libutil.....	106
V Package Format and Installation.....	107
13 Software Installation	108
13.1 Package Dependencies	108
13.2 Package Architecture Considerations	108
A Alphabetical Listing of Interfaces.....	109
A.1 libgcc_s.....	109
B GNU Free Documentation License (Informative)	110
B.1 PREAMBLE.....	110
B.2 APPLICABILITY AND DEFINITIONS.....	110
B.3 VERBATIM COPYING.....	111
B.4 COPYING IN QUANTITY	111
B.5 MODIFICATIONS	112
B.6 COMBINING DOCUMENTS.....	113
B.7 COLLECTIONS OF DOCUMENTS.....	114
B.8 AGGREGATION WITH INDEPENDENT WORKS.....	114
B.9 TRANSLATION	114
B.10 TERMINATION	114
B.11 FUTURE REVISIONS OF THIS LICENSE.....	115
B.12 How to use this License for your documents.....	115

List of Tables

2-1 Normative References	10
2-2 Other References	12
3-1 Standard Library Names	14
9-1 ELF Special Sections	26
9-2 Additional Special Sections	26
11-1 libc Definition	30
11-2 libc - RPC Function Interfaces	30
11-3 libc - System Calls Function Interfaces	31
11-4 libc - Standard I/O Function Interfaces	33
11-5 libc - Standard I/O Data Interfaces	34
11-6 libc - Signal Handling Function Interfaces	34
11-7 libc - Signal Handling Data Interfaces	35
11-8 libc - Localization Functions Function Interfaces	35
11-9 libc - Localization Functions Data Interfaces	36
11-10 libc - Socket Interface Function Interfaces	36
11-11 libc - Wide Characters Function Interfaces	36
11-12 libc - String Functions Function Interfaces	38
11-13 libc - IPC Functions Function Interfaces	39
11-14 libc - Regular Expressions Function Interfaces	39
11-15 libc - Character Type Functions Function Interfaces	39
11-16 libc - Time Manipulation Function Interfaces	40
11-17 libc - Time Manipulation Data Interfaces	40
11-18 libc - Terminal Interface Functions Function Interfaces	40
11-19 libc - System Database Interface Function Interfaces	41
11-20 libc - Language Support Function Interfaces	42
11-21 libc - Large File Support Function Interfaces	42
11-22 libc - Standard Library Function Interfaces	42
11-23 libc - Standard Library Data Interfaces	44
11-24 libm Definition	69
11-25 libm - Math Function Interfaces	70
11-26 libm - Math Data Interfaces	74
11-27 libpthread Definition	80
11-28 libpthread - Realtime Threads Function Interfaces	80
11-29 libpthread - Posix Threads Function Interfaces	81
11-30 libpthread - Thread aware versions of libc interfaces Function Interfaces	82
11-31 libgcc_s Definition	87
11-32 libgcc_s - Unwind Library Function Interfaces	87
11-33 libdl Definition	96
11-34 libdl - Dynamic Loader Function Interfaces	96
11-35 libcrypt Definition	97
11-36 libcrypt - Encryption Function Interfaces	97
12-1 libz Definition	99
12-2 libncurses Definition	100
12-3 libutil Definition	106
12-4 libutil - Utility Functions Function Interfaces	106
A-1 libgcc_s Function Interfaces	109

Foreword

1 This is version 3.1 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 supplement ("LSB-arch" or "archLSB") describing the parts of the interface that vary
9 by processor architecture. Together, the LSB-generic and the architecture-specific
10 supplement for a single hardware architecture provide a complete interface
11 specification for compiled application programs on systems that share a common
12 hardware architecture.

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

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

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

1.2 Module Specific Scope

28 This is the 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 References

2.1 Normative References

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

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

10 **Table 2-1 Normative References**

Name	Title	URL
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 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	http://www.unix.org/version3/

Name	Title	URL
	-- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004	
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
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
SUSv2	CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)	http://www.opengroup.org/publications/catalog/un.htm
SUSv2 Commands and Utilities	The Single UNIX® Specification (SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/publications/catalog/un.htm
SVID Issue 3	American Telephone and Telegraph Company, System V Interface Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989. (ISBN 0201566524)	
SVID Issue 4	System V Interface Definition, Fourth Edition	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	http://www.opengroup.org/publications/catalog/un.htm

2.2 Informative References/Bibliography

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

15 **Table 2-2 Other References**

Name	Title	URL
DWARF Debugging Information Format, Revision 2.0.0	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf
DWARF Debugging Information Format, Revision 3.0.0 (Draft)	DWARF Debugging Information Format, Revision 3.0.0 (Draft)	http://refspecs.freestandards.org/dwarf/
ISO/IEC TR14652	ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions	
ITU-T V.42	International Telecommunication Union Recommendation V.42 (2002): Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion ITUV	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Li18nux Globalization Specification	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt
PAM	Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar & R.Schemers (SunSoft)	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 1831/1832 RPC & XDR	IETF RFC 1831 & 1832	http://www.ietf.org/
RFC 1833: Binding Protocols for ONC RPC	IETF RFC 1833: Binding Protocols for ONC RPC	http://www.ietf.org/rfc/rfc1833.txt

Name	Title	URL
Version 2	Version 2	
RFC 1950: ZLIB Compressed Data Format Specification	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821: Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822: Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 791: Internet Protocol	IETF RFC 791: Internet Protocol Specification	http://www.ietf.org/rfc/rfc791.txt
RPM Package Format	RPM Package Format V3.0	http://www.rpm.org/max-rpm/s1-rpm-file-format-rpm-file-format.html
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 • A processor architecture represents a family of related processors which may not
17 have identical feature sets. The architecture specific supplement to this
18 specification for a given target processor architecture describes a minimum
19 acceptable processor. The implementation shall provide all features of this
20 processor, whether in hardware or through emulation transparent to the
21 application.
- 22 • The implementation shall be capable of executing compiled applications having
23 the format and using the system interfaces described in this document.
- 24 • The implementation shall provide libraries containing the interfaces specified by
25 this document, and shall provide a dynamic linking mechanism that allows these

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

3.3 LSB Application Conformance

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

3 Requirements

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

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

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

4 Definitions

1	For the purposes of this document, the following definitions, as specified in the
2	<i>ISO/IEC Directives, Part 2, 2001, 4th Edition</i> , apply:
3	can
4	be able to; there is a possibility of; it is possible to
5	cannot
6	be unable to; there is no possibility of; it is not possible to
7	may
8	is permitted; is allowed; is permissible
9	need not
10	it is not required that; no...is required
11	shall
12	is to; is required to; it is required that; has to; only...is permitted; it is necessary
13	shall not
14	is not allowed [permitted] [acceptable] [permissible]; is required to be not; is
15	required that...be not; is not to be
16	should
17	it is recommended that; ought to
18	should not
19	it is not recommended that; ought not to

5 Terminology

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

2 archLSB

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

6 Binary Standard

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

9 gLSB

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

12 implementation-defined

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

20 Shell Script

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

23 Source Standard

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

26 undefined

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

33 unspecified

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

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

6 Documentation Conventions

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

2 `function()`

3 the name of a function

4 **command**

5 the name of a command or utility

6 `CONSTANT`

7 a constant value

8 *parameter*

9 a parameter

10 `variable`

11 a variable

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

14 `name`

15 the name of the interface

16 `(symver)`

17 An optional symbol version identifier, if required.

18 `[refno]`

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

21 For example,

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

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

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

II Executable and Linking Format (ELF)

7 Introduction

1 Executable and Linking Format (ELF) defines the object format for compiled
2 applications. This specification supplements the information found in System V ABI
3 Update and 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

1 The ESA/390 Architecture is specified by the following documents

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

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

- 6 • additional floating point facility
- 7 • compare and move extended facility
- 8 • immediate and relative instruction facility
- 9 • string instruction facility
- 10 • square-root facility

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

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

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

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

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

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

8.1.2 Data Representation

27 LSB-conforming applications shall use the data representation as defined in Chapter
28 1 of the LINUX for S/390 ELF Application Binary Interface Supplement.

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

8.1.2.1 Byte Ordering

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

8.1.2.2 Fundamental Types

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

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

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

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

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 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 `.rela.plt`

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

9.4 Symbol Table

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

9.5 Relocation

27 LSB-conforming applications shall use Relocations as defined in Chapter 2 of the
28 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 funtionatliy.

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 Program Interpreter shall be `/lib/ld-1sb-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
[SUSv2] SUSv2
[SUSv3] ISO POSIX (2003)
[SVID.3] SVID Issue 3
13 [SVID.4] SVID Issue 4

11.2.1 RPC

11.2.1.1 Interfaces for RPC

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

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

<code>authnone_create(</code> <code>GLIBC_2.0)</code> [SVID.4]	<code>clnt_create(</code> <code>GLIBC_2.0)</code> [SVID.4]	<code>clnt_pcreateerror(</code> <code>GLIBC_2.0)</code> [SVID.4]	<code>clnt_perrno(</code> <code>GLIBC_2.0)</code> [SVID.4]
<code>clnt_perror(</code> <code>GLIBC_2.0)</code> [SVID.4]	<code>clnt_screateerror(</code> <code>GLIBC_2.0)</code> [SVID.4]	<code>clnt_sperno(</code> <code>GLIBC_2.0)</code> [SVID.4]	<code>clnt_sperror(</code> <code>GLIBC_2.0)</code> [SVID.4]
<code>key_decryptsessio</code> <code>n(</code> <code>GLIBC_2.1)</code> [SVID.3]	<code>pmap_getport(</code> <code>GLIBC_2.0)</code> [LSB]	<code>pmap_set(</code> <code>GLIBC_2.0)</code> [LSB]	<code>pmap_unset(</code> <code>GLIBC_2.0)</code> [LSB]
<code>svc_getreqset(</code> <code>GLIBC_2.0)</code>	<code>svc_register(</code> <code>GLIBC_2.0)</code>	<code>svc_run(</code> <code>GLIBC_2.0)</code>	<code>svc_sendreply(</code> <code>GLIBC_2.0)</code>

BC_2.0) [SVID.3]	C_2.0) [LSB]	0) [LSB]	IBC_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]	svtcp_create(GLIBC_2.0) [LSB]
svcdp_create(GLIBC_2.0) [LSB]	xdr_accepted_reply(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_reply(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]			

19

11.2.2 System Calls

20

11.2.2.1 Interfaces for System Calls

21

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

22

23

24

Table 11-3 libc - System Calls Function Interfaces

__fxstat(GLIBC_2.0) [LSB]	__getpgid(GLIBC_2.0) [LSB]	__lxstat(GLIBC_2.0) [LSB]	__xmknod(GLIBC_2.0) [LSB]
__xstat(GLIBC_2.0) [LSB]	access(GLIBC_2.0) [SUSv3]	acct(GLIBC_2.0) [LSB]	alarm(GLIBC_2.0) [SUSv3]
brk(GLIBC_2.0) [SUSv2]	chdir(GLIBC_2.0) [SUSv3]	chmod(GLIBC_2.0) [SUSv3]	chown(GLIBC_2.1) [SUSv3]
chroot(GLIBC_2.0)	clock(GLIBC_2.0)	close(GLIBC_2.0)	closedir(GLIBC_2.0)

) [SUSv2]	[SUSv3]	[SUSv3]	0) [SUSv3]
creat(GLIBC_2.0) [SUSv3]	dup(GLIBC_2.0) [SUSv3]	dup2(GLIBC_2.0) [SUSv3]	execl(GLIBC_2.0) [SUSv3]
execle(GLIBC_2.0) [SUSv3]	execlp(GLIBC_2.0) [SUSv3]	execv(GLIBC_2.0) [SUSv3]	execve(GLIBC_2.0) [SUSv3]
execvp(GLIBC_2.0) [SUSv3]	exit(GLIBC_2.0) [SUSv3]	fchdir(GLIBC_2.0) [SUSv3]	fchmod(GLIBC_2.0) [SUSv3]
fchown(GLIBC_2.0) [SUSv3]	fcntl(GLIBC_2.0) [LSB]	fdatasync(GLIBC_2.0) [SUSv3]	flock(GLIBC_2.0) [LSB]
fork(GLIBC_2.0) [SUSv3]	fstatvfs(GLIBC_2.1) [SUSv3]	fsync(GLIBC_2.0) [SUSv3]	ftime(GLIBC_2.0) [SUSv3]
ftruncate(GLIBC_2.0) [SUSv3]	getcontext(GLIBC_2.1) [SUSv3]	getegid(GLIBC_2.0) [SUSv3]	geteuid(GLIBC_2.0) [SUSv3]
getgid(GLIBC_2.0) [SUSv3]	getgroups(GLIBC_2.0) [SUSv3]	getitimer(GLIBC_2.0) [SUSv3]	getloadavg(GLIBC_2.2) [LSB]
getpagesize(GLIBC_2.0) [SUSv2]	getpgid(GLIBC_2.0) [SUSv3]	getpgrp(GLIBC_2.0) [SUSv3]	getpid(GLIBC_2.0) [SUSv3]
getppid(GLIBC_2.0) [SUSv3]	getpriority(GLIBC_2.0) [SUSv3]	getrlimit(GLIBC_2.2) [SUSv3]	getrusage(GLIBC_2.0) [SUSv3]
getsid(GLIBC_2.0) [SUSv3]	getuid(GLIBC_2.0) [SUSv3]	getwd(GLIBC_2.0) [SUSv3]	initgroups(GLIBC_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_getschedul	sched_rr_get_inte	sched_setparam(sched_setschedule

er(GLIBC_2.0) [SUSv3]	rval(GLIBC_2.0) [SUSv3]	GLIBC_2.0 [SUSv3]	r(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_2.2) [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]
writenv(GLIBC_2.0) [SUSv3]			

25

11.2.3 Standard I/O

26

11.2.3.1 Interfaces for Standard I/O

27

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

28

29

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

_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]

30

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]		

31

32

33

34

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

35

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

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

36

11.2.4 Signal Handling

37

11.2.4.1 Interfaces for Signal Handling

38

39

40

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

41

Table 11-6 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.1) [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]			

42

43

44

45

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

46

Table 11-7 libc - Signal Handling Data Interfaces

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

47

11.2.5 Localization Functions

48

11.2.5.1 Interfaces for Localization Functions

49

50

51

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

52

Table 11-8 libc - Localization Functions Function Interfaces

bind_textdomain_codeset(GLIBC_2.2) [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]
dcngettext(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]		

53

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

57 **Table 11-9 libc - Localization Functions Data Interfaces**

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

58

11.2.6 Socket Interface

11.2.6.1 Interfaces for Socket Interface

59 An LSB conforming implementation shall provide the architecture specific functions
 60 for Socket Interface specified in Table 11-10, with the full mandatory functionality as
 61 described in the referenced underlying specification.
 62

63 **Table 11-10 libc - Socket Interface Function Interfaces**

__h_errno_locatio n(GLIBC_2.0) [LSB]	accept(GLIBC_2.0) [SUSv3]	bind(GLIBC_2.0) [SUSv3]	bindresvport(GLI BC_2.0) [LSB]
connect(GLIBC_2. 0) [SUSv3]	gethostid(GLIBC_ 2.0) [SUSv3]	gethostname(GLI BC_2.0) [SUSv3]	getpeername(GLI BC_2.0) [SUSv3]
getsockname(GLI BC_2.0) [SUSv3]	getsockopt(GLIBC _2.0) [LSB]	if_freenameindex(GLIBC_2.1) [SUSv3]	if_indextoname(G LIBC_2.1) [SUSv3]
if_nameindex(GLI BC_2.1) [SUSv3]	if_nametoindex(G LIBC_2.1) [SUSv3]	listen(GLIBC_2.0) [SUSv3]	recv(GLIBC_2.0) [SUSv3]
recvfrom(GLIBC_ 2.0) [SUSv3]	recvmsg(GLIBC_2 .0) [SUSv3]	send(GLIBC_2.0) [SUSv3]	sendmsg(GLIBC_ 2.0) [SUSv3]
sendto(GLIBC_2.0) [SUSv3]	setsockopt(GLIBC _2.0) [LSB]	shutdown(GLIBC _2.0) [SUSv3]	socketatmark(GLIB C_2.2.4) [SUSv3]
socket(GLIBC_2.0) [SUSv3]	socketpair(GLIBC _2.0) [SUSv3]		

64

11.2.7 Wide Characters

11.2.7.1 Interfaces for Wide Characters

65 An LSB conforming implementation shall provide the architecture specific functions
 66 for Wide Characters specified in Table 11-11, with the full mandatory functionality
 67 as described in the referenced underlying specification.
 68

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

__wcstod_internal (GLIBC_2.0) [LSB]	__wcstof_internal(GLIBC_2.0) [LSB]	__wcstol_internal(GLIBC_2.0) [LSB]	__wcstold_interna l(GLIBC_2.0) [LSB]
__wcstoul_interna l(GLIBC_2.0)	btowc(GLIBC_2.0) [SUSv3]	fgetwc(GLIBC_2.2) [SUSv3]	fgetws(GLIBC_2.2) [SUSv3]

[LSB]			
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]	mbsnrtowcs(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]
wscasecmp(GLIBC_2.1) [LSB]	wscat(GLIBC_2.0) [SUSv3]	wcschr(GLIBC_2.0) [SUSv3]	wscmp(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]	wcsncasecmp(GLIBC_2.1) [LSB]	wcsncat(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]	wcstod(GLIBC_2.0) [SUSv3]	wcstof(GLIBC_2.0) [SUSv3]	wcstoimax(GLIBC_2.1) [SUSv3]
wcstok(GLIBC_2.0) [SUSv3]	wcstol(GLIBC_2.0) [SUSv3]	wcstold(GLIBC_2.0) [SUSv3]	wcstoll(GLIBC_2.1) [SUSv3]
wctombs(GLIBC_2.0) [SUSv3]	wcstoq(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]

70

wscanf(GLIBC_2.2) [LSB]			
-------------------------	--	--	--

11.2.8 String Functions

71

11.2.8.1 Interfaces for String Functions

72

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

73

74

75

Table 11-12 libc - String Functions Function Interfaces

__mempcpy(GLIBC_2.0) [LSB]	__rawmemchr(GLIBC_2.1) [LSB]	__stpcpy(GLIBC_2.0) [LSB]	__strdup(GLIBC_2.0) [LSB]
__strtod_internal(GLIBC_2.0) [LSB]	__strtof_internal(GLIBC_2.0) [LSB]	__strtok_r(GLIBC_2.0) [LSB]	__strtold_internal(GLIBC_2.0) [LSB]
__strtold_internal(GLIBC_2.0) [LSB]	__strtoll_internal(GLIBC_2.0) [LSB]	__strtoul_internal(GLIBC_2.0) [LSB]	__strtoull_internal(GLIBC_2.0) [LSB]
bcmp(GLIBC_2.0) [SUSv3]	bcopy(GLIBC_2.0) [SUSv3]	bzero(GLIBC_2.0) [SUSv3]	ffs(GLIBC_2.0) [SUSv3]
index(GLIBC_2.0) [SUSv3]	memccpy(GLIBC_2.0) [SUSv3]	memchr(GLIBC_2.0) [SUSv3]	memcmp(GLIBC_2.0) [SUSv3]
memcpy(GLIBC_2.0) [SUSv3]	memmove(GLIBC_2.0) [SUSv3]	memrchr(GLIBC_2.2) [LSB]	memset(GLIBC_2.0) [SUSv3]
rindex(GLIBC_2.0) [SUSv3]	stpcpy(GLIBC_2.0) [LSB]	stpncpy(GLIBC_2.0) [LSB]	strcasemp(GLIBC_2.0) [SUSv3]
strcasestr(GLIBC_2.1) [LSB]	strcat(GLIBC_2.0) [SUSv3]	strchr(GLIBC_2.0) [SUSv3]	strcmp(GLIBC_2.0) [SUSv3]
strcoll(GLIBC_2.0) [SUSv3]	strcpy(GLIBC_2.0) [SUSv3]	strcspn(GLIBC_2.0) [SUSv3]	strdup(GLIBC_2.0) [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]	strncasemp(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]	strrchr(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]

76

strxfrm(GLIBC_2.0) [SUSv3]	swab(GLIBC_2.0) [SUSv3]		
----------------------------	-------------------------	--	--

11.2.9 IPC Functions

77

11.2.9.1 Interfaces for IPC Functions

78

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

79

80

81

Table 11-13 libc - IPC Functions Function Interfaces

ftok(GLIBC_2.0) [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) [SUSv3]	shmctl(GLIBC_2.2) [SUSv3]	shmdt(GLIBC_2.0) [SUSv3]	shmget(GLIBC_2.0) [SUSv3]

82

11.2.10 Regular Expressions

83

11.2.10.1 Interfaces for Regular Expressions

84

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

85

86

87

Table 11-14 libc - Regular Expressions Function Interfaces

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

88

11.2.11 Character Type Functions

89

11.2.11.1 Interfaces for Character Type Functions

90

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

91

92

93

Table 11-15 libc - Character Type Functions Function Interfaces

__ctype_get_mb_cur_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]	

94

11.2.12 Time Manipulation

95

11.2.12.1 Interfaces for Time Manipulation

96

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

97

98

99

Table 11-16 libc - Time Manipulation Function Interfaces

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]	difftime(GLIBC_2.0) [SUSv3]	gmtime(GLIBC_2.0) [SUSv3]	gmtime_r(GLIBC_2.0) [SUSv3]
localtime(GLIBC_2.0) [SUSv3]	localtime_r(GLIBC_2.0) [SUSv3]	mktime(GLIBC_2.0) [SUSv3]	tzset(GLIBC_2.0) [SUSv3]
ualarm(GLIBC_2.0) [SUSv3]			

100

101

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

102

103

104

Table 11-17 libc - Time Manipulation Data Interfaces

__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]		

105

11.2.13 Terminal Interface Functions

106

11.2.13.1 Interfaces for Terminal Interface Functions

107

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

108

109

110

Table 11-18 libc - Terminal Interface Functions Function Interfaces

cfgetispeed(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]	

111

11.2.14 System Database Interface

112

11.2.14.1 Interfaces for System Database Interface

113

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

114

115

116

Table 11-19 libc - System Database Interface Function Interfaces

endgrent(GLIBC_2.0) [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]	getprotobyname_r(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]	

117

11.2.15 Language Support

118

11.2.15.1 Interfaces for Language Support

119

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

120

121

122

Table 11-20 libc - Language Support Function Interfaces

123

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

11.2.16 Large File Support

124

11.2.16.1 Interfaces for Large File Support

125

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

126

127

Table 11-21 libc - Large File Support Function Interfaces

128

__fxstat64(GLIBC_2.2) [LSB]	__lxstat64(GLIBC_2.2) [LSB]	__xstat64(GLIBC_2.2) [LSB]	creat64(GLIBC_2.1) [LFS]
fgetpos64(GLIBC_2.2) [LFS]	fopen64(GLIBC_2.1) [LFS]	freopen64(GLIBC_2.1) [LFS]	fseeko64(GLIBC_2.1) [LFS]
fsetpos64(GLIBC_2.2) [LFS]	fstatvfs64(GLIBC_2.1) [LFS]	ftello64(GLIBC_2.1) [LFS]	ftruncate64(GLIBC_2.1) [LFS]
ftw64(GLIBC_2.1) [LFS]	getrlimit64(GLIBC_2.2) [LFS]	lockf64(GLIBC_2.1) [LFS]	mkstemp64(GLIBC_2.2) [LFS]
mmap64(GLIBC_2.1) [LFS]	nftw64(GLIBC_2.3) [LFS]	readdir64(GLIBC_2.2) [LFS]	statvfs64(GLIBC_2.1) [LFS]
tmpfile64(GLIBC_2.1) [LFS]	truncate64(GLIBC_2.1) [LFS]		

129

11.2.17 Standard Library

130

11.2.17.1 Interfaces for Standard Library

131

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

132

133

134

Table 11-22 libc - Standard Library Function Interfaces

_Exit(GLIBC_2.1.1) [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]	__isinff(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)	atol(GLIBC_2.0)	atoll(GLIBC_2.0)	basename(GLIBC_2.0)

[SUSv3]	[SUSv3]	[SUSv3]	_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_2.2.3) [SUSv3]	fpathconf(GLIBC_2.0) [SUSv3]	free(GLIBC_2.0) [SUSv3]	freeaddrinfo(GLIBC_2.0) [SUSv3]
ftwlockfile(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]	ttyname(GLIBC_2.0) [SUSv3]	ttyname_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]			

135

136

137

138

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

139

Table 11-23 libc - Standard Library Data Interfaces

__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]			

140

11.3 Data Definitions for libc

141

142

143

144

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

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

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

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

11.3.1 arpa/inet.h

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

11.3.2 assert.h

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

11.3.3 ctype.h

```
167 extern int _tolower(int);
168 extern int _toupper(int);
169 extern int isalnum(int);
170 extern int isalpha(int);
171 extern int isascii(int);
172 extern int iscntrl(int);
173 extern int isdigit(int);
174 extern int isgraph(int);
175 extern int islower(int);
176 extern int isprint(int);
177 extern int ispunct(int);
178 extern int isspace(int);
179 extern int isupper(int);
180 extern int isxdigit(int);
181 extern int toascii(int);
182 extern int tolower(int);
183 extern int toupper(int);
184 extern int isblank(int);
185 extern const unsigned short **__ctype_b_loc(void);
186 extern const int32_t **__ctype_toupper_loc(void);
187 extern const int32_t **__ctype_tolower_loc(void);
188
```

11.3.4 dirent.h

```
189
190 extern void rewinddir(DIR *);
191 extern void seekdir(DIR *, long int);
192 extern long int telldir(DIR *);
```

```

193     extern int closedir(DIR *);
194     extern DIR *opendir(const char *);
195     extern struct dirent *readdir(DIR *);
196     extern struct dirent64 *readdir64(DIR *);
197     extern int readdir_r(DIR *, struct dirent *, struct dirent **);

```

11.3.5 err.h

```

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

```

11.3.6 errno.h

```

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

```

11.3.7 fcntl.h

```

208
209     #define F_GETLK64          12
210     #define F_SETLK64          13
211     #define F_SETLKW64         14
212
213     extern int lockf64(int, int, off64_t);
214     extern int fcntl(int, int, ...);

```

11.3.8 fmtmsg.h

```

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

```

11.3.9 fnmatch.h

```

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

```

11.3.10 ftw.h

```

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

```

11.3.11 getopt.h

```

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

```

11.3.12 glob.h

```

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

```

11.3.13 grp.h

```

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

```

11.3.14 iconv.h

```

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

```

11.3.15 inttypes.h

```

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

```

11.3.16 langinfo.h

```

269
270 extern char *nl_langinfo(nl_item);

```

11.3.17 libgen.h

```

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

```

11.3.18 libintl.h

```

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

```

11.3.19 limits.h

```

286
287 #define ULONG_MAX      0xFFFFFFFFUL
288 #define LONG_MAX       2147483647
289
290 #define CHAR_MIN      0
291 #define CHAR_MAX      255
292
293 #define PTHREAD_STACK_MIN 16384

```

11.3.20 locale.h

```

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

```

11.3.21 monetary.h

```

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

```

11.3.22 net/if.h

```

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

```

11.3.23 netdb.h

```

308
309 extern void endprotoent(void);
310 extern void endservent(void);
311 extern void freeaddrinfo(struct addrinfo *);
312 extern const char *gai_strerror(int);
313 extern int getaddrinfo(const char *, const char *, const struct addrinfo
314 *,
315                       struct addrinfo **);
316 extern struct hostent *gethostbyaddr(const void *, socklen_t, int);
317 extern struct hostent *gethostbyname(const char *);
318 extern struct protoent *getprotobyname(const char *);

```



```

319 extern struct protoent *getprotobyname(int);
320 extern struct protoent *getprotoent(void);
321 extern struct servent *getservbyname(const char *, const char *);
322 extern struct servent *getservbyport(int, const char *);
323 extern struct servent *getservent(void);
324 extern void setprotoent(int);
325 extern void setservent(int);
326 extern int *__h_errno_location(void);

```

11.3.24 netinet/in.h

```

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

```

11.3.25 netinet/ip.h

```

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

```

11.3.26 netinet/tcp.h

```

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

```

11.3.27 netinet/udp.h

```

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

```

11.3.28 nl_types.h

```

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

```

11.3.29 poll.h

```

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

```

11.3.30 pty.h

```

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

```

11.3.31 pwd.h

```

354
355 extern void endpwent(void);
356 extern struct passwd *getpwent(void);

```

```

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

```

11.3.32 regex.h

```

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

```

11.3.33 rpc/auth.h

```

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

```

11.3.34 rpc/clnt.h

```

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

```

11.3.35 rpc/pmap_clnt.h

```

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

```

11.3.36 rpc/rpc_msg.h

```

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

```

11.3.37 rpc/svc.h

```

391
392     extern void svc_getreqset(fd_set *);
393     extern bool_t svc_register(SVCXPRT *, rpcprog_t, rpcvers_t,
394                               __dispatch_fn_t, rpcprot_t);
395     extern void svc_run(void);
396     extern bool_t svc_sendreply(SVCXPRT *, xdrproc_t, caddr_t);
397     extern void svcerr_auth(SVCXPRT *, enum auth_stat);
398     extern void svcerr_decode(SVCXPRT *);
399     extern void svcerr_noproc(SVCXPRT *);
400     extern void svcerr_noprogram(SVCXPRT *);

```

```

401 extern void svcerr_progvers(SVCXPRT *, rpcvers_t, rpcvers_t);
402 extern void svcerr_systemerr(SVCXPRT *);
403 extern void svcerr_weakauth(SVCXPRT *);
404 extern SVCXPRT *svctcp_create(int, u_int, u_int);
405 extern SVCXPRT *sv cudp_create(int);

```

11.3.38 rpc/types.h

```

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

```

11.3.39 rpc/xdr.h

```

411
412 extern bool_t xdr_array(XDR *, caddr_t *, u_int *, u_int, u_int,
413                        xdrproc_t);
414 extern bool_t xdr_bool(XDR *, bool_t *);
415 extern bool_t xdr_bytes(XDR *, char **, u_int *, u_int);
416 extern bool_t xdr_char(XDR *, char *);
417 extern bool_t xdr_double(XDR *, double *);
418 extern bool_t xdr_enum(XDR *, enum_t *);
419 extern bool_t xdr_float(XDR *, float *);
420 extern void xdr_free(xdrproc_t, char *);
421 extern bool_t xdr_int(XDR *, int *);
422 extern bool_t xdr_long(XDR *, long int *);
423 extern bool_t xdr_opaque(XDR *, caddr_t, u_int);
424 extern bool_t xdr_pointer(XDR *, char **, u_int, xdrproc_t);
425 extern bool_t xdr_reference(XDR *, caddr_t *, u_int, xdrproc_t);
426 extern bool_t xdr_short(XDR *, short *);
427 extern bool_t xdr_string(XDR *, char **, u_int);
428 extern bool_t xdr_u_char(XDR *, u_char *);
429 extern bool_t xdr_u_int(XDR *, u_int *);
430 extern bool_t xdr_u_long(XDR *, u_long *);
431 extern bool_t xdr_u_short(XDR *, u_short *);
432 extern bool_t xdr_union(XDR *, enum_t *, char *,
433                        const struct xdr_discrim *, xdrproc_t);
434 extern bool_t xdr_vector(XDR *, char *, u_int, u_int, xdrproc_t);
435 extern bool_t xdr_void(void);
436 extern bool_t xdr_wrapstring(XDR *, char **);
437 extern void xdrmem_create(XDR *, caddr_t, u_int, enum xdr_op);
438 extern void xdrrec_create(XDR *, u_int, u_int, caddr_t,
439                          int (*__readit) (char *p1, char *p2, int p3)
440                          , int (*__writeit) (char *p1, char *p2, int
441                          p3)
442                          );
443 extern typedef int bool_t xdrrec_eof(XDR *);

```

11.3.40 sched.h

```

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

```

11.3.41 search.h

```

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

```

11.3.42 setjmp.h

```

467
468     typedef int __jmp_buf[14];
469
470     extern int __sigsetjmp(jmp_buf, int);
471     extern void longjmp(jmp_buf, int);
472     extern void siglongjmp(sigjmp_buf, int);
473     extern void _longjmp(jmp_buf, int);
474     extern int _setjmp(jmp_buf);

```

11.3.43 signal.h

```

475
476     #define __NUM_ACRS      16
477     #define __NUM_FPRS     16
478     #define __NUM_GPRS     16
479
480     typedef struct {
481         unsigned long int mask;
482         unsigned long int addr;
483     } __attribute__((aligned(8)))
484     _psw_t;
485     typedef struct {
486         _psw_t psw;
487         unsigned long int gprs[__NUM_GPRS];
488         unsigned int acrs[__NUM_ACRS];
489     } _s390_regs_common;
490
491     #define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int))-3)
492
493     #define SI_PAD_SIZE    ((SI_MAX_SIZE/sizeof(int))-3)
494
495     struct sigaction {
496         union {
497             sighandler_t _sa_handler;
498             void (*_sa_sigaction) (int, siginfo_t *, void *);
499         } __sigaction_handler;
500         sigset_t sa_mask;
501         unsigned long int sa_flags;
502         void (*sa_restorer) (void);
503     };
504
505     #define MINSIGSTKSZ    2048
506     #define SIGSTKSZ      8192
507

```

```

508     typedef struct {
509         unsigned int fpc;
510         double fprs[__NUM_FPRS];
511     } _s390_fp_regs;
512     typedef struct {
513         _s390_regs_common regs;
514         _s390_fp_regs fpregs;
515     } _sigregs;
516
517     struct sigcontext {
518         unsigned long int oldmask[2];
519         _sigregs *sregs;
520     };
521     extern int __libc_current_sigrtmax(void);
522     extern int __libc_current_sigrtmin(void);
523     extern sighandler_t __sysv_signal(int, sighandler_t);
524     extern char *const _sys_siglist(void);
525     extern int killpg(pid_t, int);
526     extern void psignal(int, const char *);
527     extern int raise(int);
528     extern int sigaddset(sigset_t *, int);
529     extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
530     extern int sigdelset(sigset_t *, int);
531     extern int sigemptyset(sigset_t *);
532     extern int sigfillset(sigset_t *);
533     extern int sighold(int);
534     extern int sigignore(int);
535     extern int siginterrupt(int, int);
536     extern int sigisemptyset(const sigset_t *);
537     extern int sigismember(const sigset_t *, int);
538     extern int sigorset(sigset_t *, const sigset_t *, const sigset_t *);
539     extern int sigpending(sigset_t *);
540     extern int sigrelse(int);
541     extern sighandler_t sigset(int, sighandler_t);
542     extern int pthread_kill(pthread_t, int);
543     extern int pthread_sigmask(int, sigset_t *, sigset_t *);
544     extern int sigaction(int, const struct sigaction *, struct sigaction *);
545     extern int sigwait(sigset_t *, int *);
546     extern int kill(pid_t, int);
547     extern int sigaltstack(const struct sigaltstack *, struct sigaltstack
548     *);
549     extern sighandler_t signal(int, sighandler_t);
550     extern int sigpause(int);
551     extern int sigprocmask(int, const sigset_t *, sigset_t *);
552     extern int sigreturn(struct sigcontext *);
553     extern int sigsuspend(const sigset_t *);
554     extern int sigqueue(pid_t, int, const union sigval);
555     extern int sigwaitinfo(const sigset_t *, siginfo_t *);
556     extern int sigtimedwait(const sigset_t *, siginfo_t *,
557         const struct timespec *);
558     extern sighandler_t bsd_signal(int, sighandler_t);

```

11.3.44 stddef.h

```

559
560     typedef unsigned long int size_t;
561     typedef int ptrdiff_t;

```

11.3.45 stdio.h

```

562
563     #define __IO_FILE_SIZE 152
564
565     extern char *const _sys_errlist(void);

```

```

566     extern void clearerr(FILE *);
567     extern int fclose(FILE *);
568     extern FILE *fdopen(int, const char *);
569     extern int fflush_unlocked(FILE *);
570     extern int fileno(FILE *);
571     extern FILE *fopen(const char *, const char *);
572     extern int fprintf(FILE *, const char *, ...);
573     extern int fputc(int, FILE *);
574     extern FILE *freopen(const char *, const char *, FILE *);
575     extern FILE *freopen64(const char *, const char *, FILE *);
576     extern int fscanf(FILE *, const char *, ...);
577     extern int fseek(FILE *, long int, int);
578     extern int fseeko(FILE *, off_t, int);
579     extern int fseeko64(FILE *, loff_t, int);
580     extern off_t ftello(FILE *);
581     extern loff_t ftello64(FILE *);
582     extern int getchar(void);
583     extern int getchar_unlocked(void);
584     extern int getw(FILE *);
585     extern int pclose(FILE *);
586     extern void perror(const char *);
587     extern FILE *popen(const char *, const char *);
588     extern int printf(const char *, ...);
589     extern int putc_unlocked(int, FILE *);
590     extern int putchar(int);
591     extern int putchar_unlocked(int);
592     extern int putw(int, FILE *);
593     extern int remove(const char *);
594     extern void rewind(FILE *);
595     extern int scanf(const char *, ...);
596     extern void setbuf(FILE *, char *);
597     extern int sprintf(char *, const char *, ...);
598     extern int sscanf(const char *, const char *, ...);
599     extern FILE *stderr(void);
600     extern FILE *stdin(void);
601     extern FILE *stdout(void);
602     extern char *tempnam(const char *, const char *);
603     extern FILE *tmpfile64(void);
604     extern FILE *tmpfile(void);
605     extern char *tmpnam(char *);
606     extern int vfprintf(FILE *, const char *, va_list);
607     extern int vprintf(const char *, va_list);
608     extern int feof(FILE *);
609     extern int ferror(FILE *);
610     extern int fflush(FILE *);
611     extern int fgetc(FILE *);
612     extern int fgetpos(FILE *, fpos_t *);
613     extern char *fgets(char *, int, FILE *);
614     extern int fputs(const char *, FILE *);
615     extern size_t fread(void *, size_t, size_t, FILE *);
616     extern int fsetpos(FILE *, const fpos_t *);
617     extern long int ftell(FILE *);
618     extern size_t fwrite(const void *, size_t, size_t, FILE *);
619     extern int getc(FILE *);
620     extern int putc(int, FILE *);
621     extern int puts(const char *);
622     extern int setvbuf(FILE *, char *, int, size_t);
623     extern int snprintf(char *, size_t, const char *, ...);
624     extern int ungetc(int, FILE *);
625     extern int vsnprintf(char *, size_t, const char *, va_list);
626     extern int vsprintf(char *, const char *, va_list);
627     extern void flockfile(FILE *);
628     extern int asprintf(char **, const char *, ...);
629     extern int fgetpos64(FILE *, fpos64_t *);

```

```

630 extern FILE *fopen64(const char *, const char *);
631 extern int fsetpos64(FILE *, const fpos64_t *);
632 extern int ftrylockfile(FILE *);
633 extern void funlockfile(FILE *);
634 extern int getc_unlocked(FILE *);
635 extern void setbuffer(FILE *, char *, size_t);
636 extern int vasprintf(char **, const char *, va_list);
637 extern int vdprintf(int, const char *, va_list);
638 extern int vfscanf(FILE *, const char *, va_list);
639 extern int vscanf(const char *, va_list);
640 extern int vsscanf(const char *, const char *, va_list);
641 extern size_t __fpending(FILE *);

```

11.3.46 stdlib.h

```

642
643 extern double __strtod_internal(const char *, char **, int);
644 extern float __strtof_internal(const char *, char **, int);
645 extern long int __strtol_internal(const char *, char **, int, int);
646 extern long double __strtold_internal(const char *, char **, int);
647 extern long long int __strtoll_internal(const char *, char **, int, int);
648 extern unsigned long int __strtoul_internal(const char *, char **, int,
649                                             int);
650 extern unsigned long long int __strtoull_internal(const char *, char **,
651                                                  int, int);
652
653 extern long int a64l(const char *);
654 extern void abort(void);
655 extern int abs(int);
656 extern double atof(const char *);
657 extern int atoi(char *);
658 extern long int atol(char *);
659 extern long long int atoll(const char *);
660 extern void *bsearch(const void *, const void *, size_t, size_t,
661                     __compar_fn_t);
662
663 extern div_t div(int, int);
664 extern double drand48(void);
665 extern char *ecvt(double, int, int *, int *);
666 extern double erand48(unsigned short);
667 extern void exit(int);
668 extern char *fcvt(double, int, int *, int *);
669 extern char *gcvt(double, int, char *);
670 extern char *getenv(const char *);
671 extern int getsuopt(char **, char *const *, char **);
672 extern int grantpt(int);
673 extern long int jrand48(unsigned short);
674 extern char *l64a(long int);
675 extern long int labs(long int);
676 extern void lcong48(unsigned short);
677 extern ldiv_t ldiv(long int, long int);
678 extern long long int llabs(long long int);
679 extern lldiv_t lldiv(long long int, long long int);
680 extern long int lrand48(void);
681 extern int mblen(const char *, size_t);
682 extern size_t mbstowcs(wchar_t *, const char *, size_t);
683 extern int mbtowc(wchar_t *, const char *, size_t);
684 extern char *mktemp(char *);
685 extern long int mrand48(void);
686 extern long int nrand48(unsigned short);
687 extern char *ptsname(int);
688 extern int putenv(char *);
689 extern void qsort(void *, size_t, size_t, __compar_fn_t);
690 extern int rand(void);
691 extern int rand_r(unsigned int *);
692 extern unsigned short *seed48(unsigned short);

```

```

691     extern void srand48(long int);
692     extern int unlockpt(int);
693     extern size_t wcstombs(char *, const wchar_t *, size_t);
694     extern int wctomb(char *, wchar_t);
695     extern int system(const char *);
696     extern void *calloc(size_t, size_t);
697     extern void free(void *);
698     extern char *initstate(unsigned int, char *, size_t);
699     extern void *malloc(size_t);
700     extern long int random(void);
701     extern void *realloc(void *, size_t);
702     extern char *setstate(char *);
703     extern void srand(unsigned int);
704     extern void srandom(unsigned int);
705     extern double strtod(char *, char **);
706     extern float strtof(const char *, char **);
707     extern long int strtol(char *, char **, int);
708     extern long double strtold(const char *, char **);
709     extern long long int strtoll(const char *, char **, int);
710     extern long long int strtoll(const char *, char **, int);
711     extern unsigned long int strtoul(const char *, char **, int);
712     extern unsigned long long int strtoull(const char *, char **, int);
713     extern unsigned long long int strtouq(const char *, char **, int);
714     extern void _Exit(int);
715     extern size_t __ctype_get_mb_cur_max(void);
716     extern char **environ(void);
717     extern char *realpath(const char *, char *);
718     extern int setenv(const char *, const char *, int);
719     extern int unsetenv(const char *);
720     extern int getloadavg(double, int);
721     extern int mkstemp64(char *);
722     extern int posix_memalign(void **, size_t, size_t);
723     extern int posix_openpt(int);

```

11.3.47 string.h

```

724     extern void *__memcpy(void *, const void *, size_t);
725     extern char *__strcpy(char *, const char *);
726     extern char *__strtok_r(char *, const char *, char **);
727     extern void bcopy(void *, void *, size_t);
728     extern void *memchr(void *, int, size_t);
729     extern int memcmp(void *, void *, size_t);
730     extern void *memcpy(void *, void *, size_t);
731     extern void *memmem(const void *, size_t, const void *, size_t);
732     extern void *memmove(void *, const void *, size_t);
733     extern void *memset(void *, int, size_t);
734     extern char *strcat(char *, const char *);
735     extern char *strchr(char *, int);
736     extern int strcmp(char *, char *);
737     extern int strcoll(const char *, const char *);
738     extern char *strcpy(char *, char *);
739     extern size_t strcspn(const char *, const char *);
740     extern char *strerror(int);
741     extern size_t strlen(char *);
742     extern char *strncat(char *, char *, size_t);
743     extern int strncmp(char *, char *, size_t);
744     extern char *strncpy(char *, char *, size_t);
745     extern char *strpbrk(const char *, const char *);
746     extern char *strrchr(char *, int);
747     extern char *strsignal(int);
748     extern size_t strspn(const char *, const char *);
749     extern char *strstr(char *, char *);
750     extern char *strtok(char *, const char *);
751

```



```

752     extern size_t strxfrm(char *, const char *, size_t);
753     extern int bcmp(void *, void *, size_t);
754     extern void bzero(void *, size_t);
755     extern int ffs(int);
756     extern char *index(char *, int);
757     extern void *memccpy(void *, const void *, int, size_t);
758     extern char *rindex(char *, int);
759     extern int strcasecmp(char *, char *);
760     extern char *strdup(char *);
761     extern int strncasecmp(char *, char *, size_t);
762     extern char *strndup(const char *, size_t);
763     extern size_t strlen(const char *, size_t);
764     extern char *strsep(char **, const char *);
765     extern char *strerror_r(int, char *, size_t);
766     extern char *strtok_r(char *, const char *, char **);
767     extern char *strcasestr(const char *, const char *);
768     extern char *stpcpy(char *, const char *);
769     extern char *stpncpy(char *, const char *, size_t);
770     extern void *memrchr(const void *, int, size_t);

```

11.3.48 sys/file.h

```

771
772     extern int flock(int, int);

```

11.3.49 sys/ioctl.h

```

773
774     #define TIOCGWINSZ      0x5413
775     #define FIONREAD       0x541B
776     #define TIOCNOTTY     21538
777
778     extern int ioctl(int, unsigned long int, ...);

```

11.3.50 sys/ipc.h

```

779
780     struct ipc_perm {
781         key_t __key;
782         uid_t uid;
783         gid_t gid;
784         uid_t cuid;
785         uid_t cgid;
786         unsigned short mode;
787         unsigned short __pad1;
788         unsigned short __seq;
789         unsigned short __pad2;
790         unsigned long int __unused1;
791         unsigned long int __unused2;
792     };
793
794     extern key_t ftok(char *, int);

```

11.3.51 sys/mman.h

```

795
796     #define MCL_CURRENT     1
797     #define MCL_FUTURE     2
798
799     extern int msync(void *, size_t, int);
800     extern int mlock(const void *, size_t);
801     extern int mlockall(int);
802     extern void *mmap(void *, size_t, int, int, int, off_t);

```

```

803     extern int mprotect(void *, size_t, int);
804     extern int munlock(const void *, size_t);
805     extern int munlockall(void);
806     extern int munmap(void *, size_t);
807     extern void *mmap64(void *, size_t, int, int, int, off64_t);
808     extern int shm_open(const char *, int, mode_t);
809     extern int shm_unlink(const char *);

```

11.3.52 sys/msg.h

```

810
811     typedef unsigned long int msglen_t;
812     typedef unsigned long int msgqnum_t;
813
814     struct msqid_ds {
815         struct ipc_perm msg_perm;
816         time_t msg_stime;
817         unsigned long int __unused1;
818         time_t msg_rtime;
819         unsigned long int __unused2;
820         time_t msg_ctime;
821         unsigned long int __unused3;
822         unsigned long int __msg_cbytes;
823         msgqnum_t msg_qnum;
824         msglen_t msg_qbytes;
825         pid_t msg_lspid;
826         pid_t msg_lrpid;
827         unsigned long int __unused4;
828         unsigned long int __unused5;
829     };
830     extern int msgctl(int, int, struct msqid_ds *);
831     extern int msgget(key_t, int);
832     extern int msgrcv(int, void *, size_t, long int, int);
833     extern int msgsnd(int, const void *, size_t, int);

```

11.3.53 sys/param.h

```

834
835     /*
836     * This header is architecture neutral
837     * Please refer to the generic specification for details
838     */

```

11.3.54 sys/poll.h

```

839
840     /*
841     * This header is architecture neutral
842     * Please refer to the generic specification for details
843     */

```

11.3.55 sys/resource.h

```

844
845     extern int getpriority(__priority_which_t, id_t);
846     extern int getrlimit64(id_t, struct rlimit64 *);
847     extern int setpriority(__priority_which_t, id_t, int);
848     extern int setrlimit(__rlimit_resource_t, const struct rlimit *);
849     extern int setrlimit64(__rlimit_resource_t, const struct rlimit64 *);
850     extern int getrlimit(__rlimit_resource_t, struct rlimit *);
851     extern int getrusage(int, struct rusage *);

```

11.3.56 sys/sem.h

```

852
853 struct semid_ds {
854     struct ipc_perm sem_perm;
855     time_t sem_otime;
856     unsigned long int __unused1;
857     time_t sem_ctime;
858     unsigned long int __unused2;
859     unsigned long int sem_nsems;
860     unsigned long int __unused3;
861     unsigned long int __unused4;
862 };
863 extern int semctl(int, int, int, ...);
864 extern int semget(key_t, int, int);
865 extern int semop(int, struct sembuf *, size_t);

```

11.3.57 sys/shm.h

```

866
867 #define SHMLBA (__getpagesize())
868
869 typedef unsigned long int shmatt_t;
870
871 struct shmid_ds {
872     struct ipc_perm shm_perm;
873     size_t shm_segsz;
874     time_t shm_atime;
875     unsigned long int __unused1;
876     time_t shm_dtime;
877     unsigned long int __unused2;
878     time_t shm_ctime;
879     unsigned long int __unused3;
880     pid_t shm_cpid;
881     pid_t shm_lpid;
882     shmatt_t shm_nattch;
883     unsigned long int __unused4;
884     unsigned long int __unused5;
885 };
886 extern int __getpagesize(void);
887 extern void *shmat(int, const void *, int);
888 extern int shmctl(int, int, struct shmid_ds *);
889 extern int shmdt(const void *);
890 extern int shmget(key_t, size_t, int);

```

11.3.58 sys/socket.h

```

891
892 typedef uint32_t __ss_aligntype;
893
894 #define SO_RCVLOWAT 18
895 #define SO_SNDLOWAT 19
896 #define SO_RCVTIMEO 20
897 #define SO_SNDTIMEO 21
898
899 extern int bind(int, const struct sockaddr *, socklen_t);
900 extern int getnameinfo(const struct sockaddr *, socklen_t, char *,
901                       socklen_t, char *, socklen_t, unsigned int);
902 extern int getsockname(int, struct sockaddr *, socklen_t *);
903 extern int listen(int, int);
904 extern int setsockopt(int, int, int, const void *, socklen_t);
905 extern int accept(int, struct sockaddr *, socklen_t *);
906 extern int connect(int, const struct sockaddr *, socklen_t);

```

```

907     extern ssize_t recv(int, void *, size_t, int);
908     extern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *,
909                             socklen_t *);
910     extern ssize_t recvmsg(int, struct msghdr *, int);
911     extern ssize_t send(int, const void *, size_t, int);
912     extern ssize_t sendmsg(int, const struct msghdr *, int);
913     extern ssize_t sendto(int, const void *, size_t, int,
914                           const struct sockaddr *, socklen_t);
915     extern int getpeername(int, struct sockaddr *, socklen_t *);
916     extern int getsockopt(int, int, int, void *, socklen_t *);
917     extern int shutdown(int, int);
918     extern int socket(int, int, int);
919     extern int socketpair(int, int, int, int);
920     extern int socketatmark(int);

```

11.3.59 sys/stat.h

```

921
922     #define _STAT_VER          3
923
924     struct stat {
925         dev_t st_dev;
926         unsigned int __pad1;
927         ino_t st_ino;
928         mode_t st_mode;
929         nlink_t st_nlink;
930         uid_t st_uid;
931         gid_t st_gid;
932         dev_t st_rdev;
933         unsigned int __pad2;
934         off_t st_size;
935         blksize_t st_blksize;
936         blkcnt_t st_blocks;
937         struct timespec st_atim;
938         struct timespec st_mtim;
939         struct timespec st_ctim;
940         unsigned long int __unused4;
941         unsigned long int __unused5;
942     };
943     struct stat64 {
944         dev_t st_dev;
945         int __pad1;
946         ino_t __st_ino;
947         mode_t st_mode;
948         nlink_t st_nlink;
949         uid_t st_uid;
950         gid_t st_gid;
951         dev_t st_rdev;
952         int __pad2;
953         off64_t st_size;
954         blksize_t st_blksize;
955         blkcnt64_t st_blocks;
956         struct timespec st_atim;
957         struct timespec st_mtim;
958         struct timespec st_ctim;
959         ino64_t st_ino;
960     };
961
962     extern int __fxstat(int, int, struct stat *);
963     extern int __fxstat64(int, int, struct stat64 *);
964     extern int __lxstat(int, char *, struct stat *);
965     extern int __lxstat64(int, const char *, struct stat64 *);
966     extern int __xmknod(int, const char *, mode_t, dev_t *);
967     extern int __xstat(int, const char *, struct stat *);

```

```

968 extern int __xstat64(int, const char *, struct stat64 *);
969 extern int mkfifo(const char *, mode_t);
970 extern int chmod(const char *, mode_t);
971 extern int fchmod(int, mode_t);
972 extern mode_t umask(mode_t);

```

11.3.60 sys/statvfs.h

```

973
974 struct statvfs {
975     unsigned long int f_bsize;
976     unsigned long int f_frsize;
977     fsblkcnt_t f_blocks;
978     fsblkcnt_t f_bfree;
979     fsblkcnt_t f_bavail;
980     fsfilcnt_t f_files;
981     fsfilcnt_t f_ffree;
982     fsfilcnt_t f_favail;
983     unsigned long int f_fsid;
984     int __f_unused;
985     unsigned long int f_flag;
986     unsigned long int f_namemax;
987     int __f_spare[6];
988 };
989 struct statvfs64 {
990     unsigned long int f_bsize;
991     unsigned long int f_frsize;
992     fsblkcnt64_t f_blocks;
993     fsblkcnt64_t f_bfree;
994     fsblkcnt64_t f_bavail;
995     fsfilcnt64_t f_files;
996     fsfilcnt64_t f_ffree;
997     fsfilcnt64_t f_favail;
998     unsigned long int f_fsid;
999     int __f_unused;
1000     unsigned long int f_flag;
1001     unsigned long int f_namemax;
1002     int __f_spare[6];
1003 };
1004 extern int fstatvfs(int, struct statvfs *);
1005 extern int fstatvfs64(int, struct statvfs64 *);
1006 extern int statvfs(const char *, struct statvfs *);
1007 extern int statvfs64(const char *, struct statvfs64 *);

```

11.3.61 sys/time.h

```

1008
1009 extern int getitimer(__itimer_which_t, struct itimerval *);
1010 extern int setitimer(__itimer_which_t, const struct itimerval *,
1011                     struct itimerval *);
1012 extern int adjtime(const struct timeval *, struct timeval *);
1013 extern int gettimeofday(struct timeval *, struct timezone *);
1014 extern int utimes(const char *, const struct timeval *);

```

11.3.62 sys/timeb.h

```

1015
1016 extern int ftime(struct timeb *);

```

11.3.63 sys/times.h

```

1017
1018 extern clock_t times(struct tms *);

```

11.3.64 sys/types.h

```

1019
1020     typedef long long int int64_t;
1021
1022     typedef int32_t ssize_t;
1023
1024     #define __FDSET_LONGS    32

```

11.3.65 sys/uio.h

```

1025
1026     extern ssize_t readv(int, const struct iovec *, int);
1027     extern ssize_t writev(int, const struct iovec *, int);

```

11.3.66 sys/un.h

```

1028
1029     /*
1030     * This header is architecture neutral
1031     * Please refer to the generic specification for details
1032     */

```

11.3.67 sys/utsname.h

```

1033
1034     extern int uname(struct utsname *);

```

11.3.68 sys/wait.h

```

1035
1036     extern pid_t wait(int *);
1037     extern pid_t waitpid(pid_t, int *, int);
1038     extern pid_t wait4(pid_t, int *, int, struct rusage *);

```

11.3.69 syslog.h

```

1039
1040     extern void closelog(void);
1041     extern void openlog(const char *, int, int);
1042     extern int setlogmask(int);
1043     extern void syslog(int, const char *, ...);
1044     extern void vsyslog(int, const char *, va_list);

```

11.3.70 termios.h

```

1045
1046     #define OLCUC    0000002
1047     #define ONLCR   0000004
1048     #define XCASE   0000004
1049     #define NLDLY   0000400
1050     #define CR1     0001000
1051     #define IUCLC   0001000
1052     #define CR2     0002000
1053     #define CR3     0003000
1054     #define CRDLY   0003000
1055     #define TAB1    0004000
1056     #define TAB2    0010000
1057     #define TAB3    0014000
1058     #define TABDLY  0014000
1059     #define BS1     0020000
1060     #define BSDLY   0020000

```

```

1061      #define VT1      0040000
1062      #define VTDLY    0040000
1063      #define FF1      0100000
1064      #define FFDLY    0100000
1065
1066      #define VSUSP     10
1067      #define VEOL     11
1068      #define VREPRINT 12
1069      #define VDISCARD 13
1070      #define VWERASE  14
1071      #define VEOL2    16
1072      #define VMIN     6
1073      #define VSWTC    7
1074      #define VSTART   8
1075      #define VSTOP    9
1076
1077      #define IXON      0002000
1078      #define IXOFF    0010000
1079
1080      #define CS6       0000020
1081      #define CS7       0000040
1082      #define CS8       0000060
1083      #define CSIZE     0000060
1084      #define CSTOPB    0000100
1085      #define CREAD     0000200
1086      #define PARENB    0000400
1087      #define PARODD    0001000
1088      #define HUPCL     0002000
1089      #define CLOCAL    0004000
1090      #define VTIME     5
1091
1092      #define ISIG       0000001
1093      #define ICANON    0000002
1094      #define ECHOE     0000020
1095      #define ECHOK     0000040
1096      #define ECHONL   0000100
1097      #define NOFLSH    0000200
1098      #define TOSTOP    0000400
1099      #define ECHOCTL   0001000
1100      #define ECHOPRT  0002000
1101      #define ECHOKE    0004000
1102      #define FLUSHO    0010000
1103      #define PENDIN    0040000
1104      #define IEXTEN    0100000
1105
1106      extern speed_t cfgetispeed(const struct termios *);
1107      extern speed_t cfgetospeed(const struct termios *);
1108      extern void cfmakeraw(struct termios *);
1109      extern int cfsetispeed(struct termios *, speed_t);
1110      extern int cfsetospeed(struct termios *, speed_t);
1111      extern int cfsetspeed(struct termios *, speed_t);
1112      extern int tcflow(int, int);
1113      extern int tcflush(int, int);
1114      extern pid_t tcgetsid(int);
1115      extern int tcsendbreak(int, int);
1116      extern int tcsetattr(int, int, const struct termios *);
1117      extern int tcdrain(int);
1118      extern int tcgetattr(int, struct termios *);

```

11.3.71 time.h

```

1119      extern int __daylight(void);
1120      extern long int __timezone(void);
1121

```

```

1122     extern char *__tzname(void);
1123     extern char *asctime(const struct tm *);
1124     extern clock_t clock(void);
1125     extern char *ctime(const time_t *);
1126     extern char *ctime_r(const time_t *, char *);
1127     extern double difftime(time_t, time_t);
1128     extern struct tm *getdate(const char *);
1129     extern int getdate_err(void);
1130     extern struct tm *gmtime(const time_t *);
1131     extern struct tm *localtime(const time_t *);
1132     extern time_t mktime(struct tm *);
1133     extern int stime(const time_t *);
1134     extern size_t strftime(char *, size_t, const char *, const struct tm *);
1135     extern char *strptime(const char *, const char *, struct tm *);
1136     extern time_t time(time_t *);
1137     extern int nanosleep(const struct timespec *, struct timespec *);
1138     extern int daylight(void);
1139     extern long int timezone(void);
1140     extern char *tzname(void);
1141     extern void tzset(void);
1142     extern char *asctime_r(const struct tm *, char *);
1143     extern struct tm *gmtime_r(const time_t *, struct tm *);
1144     extern struct tm *localtime_r(const time_t *, struct tm *);
1145     extern int clock_getcpuclockid(pid_t, clockid_t *);
1146     extern int clock_getres(clockid_t, struct timespec *);
1147     extern int clock_gettime(clockid_t, struct timespec *);
1148     extern int clock_nanosleep(clockid_t, int, const struct timespec *,
1149                               struct timespec *);
1150     extern int clock_settime(clockid_t, const struct timespec *);
1151     extern int timer_create(clockid_t, struct sigevent *, timer_t *);
1152     extern int timer_delete(timer_t);
1153     extern int timer_getoverrun(timer_t);
1154     extern int timer_gettime(timer_t, struct itimerspec *);
1155     extern int timer_settime(timer_t, int, const struct itimerspec *,
1156                               struct itimerspec *);

```

11.3.72 ucontext.h

```

1157
1158     #define NGREG    36
1159
1160     typedef union {
1161         double d;
1162         float f;
1163     } fpreg_t;
1164
1165     typedef struct {
1166         unsigned int fpc;
1167         fpreg_t fprs[16];
1168     } fpregset_t;
1169
1170     typedef struct {
1171         _psw_t psw;
1172         unsigned long int gregs[16];
1173         unsigned int aregs[16];
1174         fpregset_t fpregs;
1175     } mcontext_t;
1176
1177     typedef struct ucontext {
1178         unsigned long int uc_flags;
1179         struct ucontext *uc_link;
1180         stack_t uc_stack;
1181         mcontext_t uc_mcontext;
1182         sigset_t uc_sigmask;

```



```

1183     } ucontext_t;
1184 extern int getcontext(ucontext_t *);
1185 extern int makecontext(ucontext_t *, void (*func) (void)
1186                      , int, ...);
1187 extern int setcontext(const struct ucontext *);
1188 extern int swapcontext(ucontext_t *, const struct ucontext *);

```

11.3.73 ulimit.h

```

1189
1190 extern long int ulimit(int, ...);

```

11.3.74 unistd.h

```

1191
1192 typedef int intptr_t;
1193
1194 extern char **__environ(void);
1195 extern pid_t __getpgid(pid_t);
1196 extern void _exit(int);
1197 extern int acct(const char *);
1198 extern unsigned int alarm(unsigned int);
1199 extern int chown(const char *, uid_t, gid_t);
1200 extern int chroot(const char *);
1201 extern size_t confstr(int, char *, size_t);
1202 extern int creat(const char *, mode_t);
1203 extern int creat64(const char *, mode_t);
1204 extern char *ctermid(char *);
1205 extern char *cuserid(char *);
1206 extern int daemon(int, int);
1207 extern int execl(const char *, const char *, ...);
1208 extern int execlp(const char *, const char *, ...);
1209 extern int execlp(const char *, const char *, ...);
1210 extern int execv(const char *, char *const);
1211 extern int execvp(const char *, char *const);
1212 extern int fdatsync(int);
1213 extern int ftruncate64(int, off64_t);
1214 extern long int gethostid(void);
1215 extern char *getlogin(void);
1216 extern int getlogin_r(char *, size_t);
1217 extern int getopt(int, char *const, const char *);
1218 extern pid_t getpgrp(void);
1219 extern pid_t getsid(pid_t);
1220 extern char *getwd(char *);
1221 extern int lockf(int, int, off_t);
1222 extern int mkstemp(char *);
1223 extern int nice(int);
1224 extern char *optarg(void);
1225 extern int opterr(void);
1226 extern int optind(void);
1227 extern int optopt(void);
1228 extern int rename(const char *, const char *);
1229 extern int setegid(gid_t);
1230 extern int seteuid(uid_t);
1231 extern int sethostname(const char *, size_t);
1232 extern int setpgrp(void);
1233 extern void swab(const void *, void *, ssize_t);
1234 extern void sync(void);
1235 extern pid_t tcgetpgrp(int);
1236 extern int tcsetpgrp(int, pid_t);
1237 extern int truncate(const char *, off_t);
1238 extern int truncate64(const char *, off64_t);
1239 extern char *ttyname(int);
1240 extern unsigned int ualarm(useconds_t, useconds_t);

```

```

1241     extern int usleep(useconds_t);
1242     extern int close(int);
1243     extern int fsync(int);
1244     extern off_t lseek(int, off_t, int);
1245     extern int open(const char *, int, ...);
1246     extern int pause(void);
1247     extern ssize_t read(int, void *, size_t);
1248     extern ssize_t write(int, const void *, size_t);
1249     extern char *crypt(char *, char *);
1250     extern void encrypt(char *, int);
1251     extern void setkey(const char *);
1252     extern int access(const char *, int);
1253     extern int brk(void *);
1254     extern int chdir(const char *);
1255     extern int dup(int);
1256     extern int dup2(int, int);
1257     extern int execve(const char *, char *const, char *const);
1258     extern int fchdir(int);
1259     extern int fchown(int, uid_t, gid_t);
1260     extern pid_t fork(void);
1261     extern gid_t getegid(void);
1262     extern uid_t geteuid(void);
1263     extern gid_t getgid(void);
1264     extern int getgroups(int, gid_t);
1265     extern int gethostname(char *, size_t);
1266     extern pid_t getpgid(pid_t);
1267     extern pid_t getpid(void);
1268     extern uid_t getuid(void);
1269     extern int lchown(const char *, uid_t, gid_t);
1270     extern int link(const char *, const char *);
1271     extern int mkdir(const char *, mode_t);
1272     extern long int pathconf(const char *, int);
1273     extern int pipe(int);
1274     extern int readlink(const char *, char *, size_t);
1275     extern int rmdir(const char *);
1276     extern void *sbrk(ptrdiff_t);
1277     extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
1278     extern int setgid(gid_t);
1279     extern int setpgid(pid_t, pid_t);
1280     extern int setregid(gid_t, gid_t);
1281     extern int setreuid(uid_t, uid_t);
1282     extern pid_t setsid(void);
1283     extern int setuid(uid_t);
1284     extern unsigned int sleep(unsigned int);
1285     extern int symlink(const char *, const char *);
1286     extern long int sysconf(int);
1287     extern int unlink(const char *);
1288     extern pid_t vfork(void);
1289     extern ssize_t pread(int, void *, size_t, off_t);
1290     extern ssize_t pwrite(int, const void *, size_t, off_t);
1291     extern char **_environ(void);
1292     extern long int fpathconf(int, int);
1293     extern int ftruncate(int, off_t);
1294     extern char *getcwd(char *, size_t);
1295     extern int getpagesize(void);
1296     extern pid_t getppid(void);
1297     extern int isatty(int);
1298     extern loff_t lseek64(int, loff_t, int);
1299     extern int open64(const char *, int, ...);
1300     extern ssize_t pread64(int, void *, size_t, off64_t);
1301     extern ssize_t pwrite64(int, const void *, size_t, off64_t);
1302     extern int ttyname_r(int, char *, size_t);

```

11.3.75 utime.h

```

1303
1304     extern int utime(const char *, const struct utimbuf *);

```

11.3.76 utmp.h

```

1305
1306     struct lastlog {
1307         time_t ll_time;
1308         char ll_line[UT_LINESIZE];
1309         char ll_host[UT_HOSTSIZE];
1310     };
1311
1312     struct utmp {
1313         short ut_type;
1314         pid_t ut_pid;
1315         char ut_line[UT_LINESIZE];
1316         char ut_id[4];
1317         char ut_user[UT_NAMESIZE];
1318         char ut_host[UT_HOSTSIZE];
1319         struct exit_status ut_exit;
1320         long int ut_session;
1321         struct timeval ut_tv;
1322         int32_t ut_addr_v6[4];
1323         char __unused[20];
1324     };
1325
1326     extern void endutent(void);
1327     extern struct utmp *getutent(void);
1328     extern void setutent(void);
1329     extern int getutent_r(struct utmp *, struct utmp **);
1330     extern int utmpname(const char *);
1331     extern int login_tty(int);
1332     extern void login(const struct utmp *);
1333     extern int logout(const char *);
1334     extern void logwtmp(const char *, const char *, const char *);

```

11.3.77 utmpx.h

```

1335
1336     struct utmpx {
1337         short ut_type;
1338         pid_t ut_pid;
1339         char ut_line[UT_LINESIZE];
1340         char ut_id[4];
1341         char ut_user[UT_NAMESIZE];
1342         char ut_host[UT_HOSTSIZE];
1343         struct exit_status ut_exit;
1344         long int ut_session;
1345         struct timeval ut_tv;
1346         int32_t ut_addr_v6[4];
1347         char __unused[20];
1348     };
1349
1350     extern void endutxent(void);
1351     extern struct utmpx *getutxent(void);
1352     extern struct utmpx *getutxid(const struct utmpx *);
1353     extern struct utmpx *getutxline(const struct utmpx *);
1354     extern struct utmpx *pututxline(const struct utmpx *);
1355     extern void setutxent(void);

```

11.3.78 wchar.h

```

1356
1357 extern double __wcstod_internal(const wchar_t *, wchar_t **, int);
1358 extern float __wcstof_internal(const wchar_t *, wchar_t **, int);
1359 extern long int __wcstol_internal(const wchar_t *, wchar_t **, int,
1360 int);
1361 extern long double __wcstold_internal(const wchar_t *, wchar_t **, int);
1362 extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t *
1363 *,
1364 int, int);
1365 extern wchar_t *wscat(wchar_t *, const wchar_t *);
1366 extern wchar_t *wcschr(const wchar_t *, wchar_t);
1367 extern int wcsncmp(const wchar_t *, const wchar_t *);
1368 extern int wscoll(const wchar_t *, const wchar_t *);
1369 extern wchar_t *wscpy(wchar_t *, const wchar_t *);
1370 extern size_t wcsncpy(const wchar_t *, const wchar_t *);
1371 extern wchar_t *wcsdup(const wchar_t *);
1372 extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
1373 extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
1374 extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
1375 extern wchar_t *wcpbrk(const wchar_t *, const wchar_t *);
1376 extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
1377 extern size_t wcsspn(const wchar_t *, const wchar_t *);
1378 extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
1379 extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t **);
1380 extern int wcswidth(const wchar_t *, size_t);
1381 extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
1382 extern int wctob(wint_t);
1383 extern int wcwidth(wchar_t);
1384 extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
1385 extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
1386 extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
1387 extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
1388 extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
1389 extern size_t mbrlen(const char *, size_t, mbstate_t *);
1390 extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
1391 extern int mbsinit(const mbstate_t *);
1392 extern size_t mbsnrtowcs(wchar_t *, const char **, size_t, size_t,
1393 mbstate_t *);
1394 extern size_t mbsrtowcs(wchar_t *, const char **, size_t, mbstate_t *);
1395 extern wchar_t *wcpncpy(wchar_t *, const wchar_t *);
1396 extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
1397 extern size_t wcrtombs(char *, wchar_t, mbstate_t *);
1398 extern size_t wcslen(const wchar_t *);
1399 extern size_t wcsnrtombs(char *, const wchar_t **, size_t, size_t,
1400 mbstate_t *);
1401 extern size_t wcsrtombs(char *, const wchar_t **, size_t, mbstate_t *);
1402 extern double wcstod(const wchar_t *, wchar_t **);
1403 extern float wcstof(const wchar_t *, wchar_t **);
1404 extern long int wcstol(const wchar_t *, wchar_t **, int);
1405 extern long double wcstold(const wchar_t *, wchar_t **);
1406 extern long long int wcstoq(const wchar_t *, wchar_t **, int);
1407 extern unsigned long int wcstoul(const wchar_t *, wchar_t **, int);
1408 extern unsigned long long int wcstouq(const wchar_t *, wchar_t **, int);
1409 extern wchar_t *wswcs(const wchar_t *, const wchar_t *);
1410 extern int wscasecmp(const wchar_t *, const wchar_t *);
1411 extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
1412 extern size_t wcsnlen(const wchar_t *, size_t);
1413 extern long long int wcstoll(const wchar_t *, wchar_t **, int);
1414 extern unsigned long long int wcstoull(const wchar_t *, wchar_t **, int);
1415 extern wint_t btowc(int);
1416 extern wint_t fgetwc(FILE *);
1417 extern wint_t fgetwc_unlocked(FILE *);

```

```

1418 extern wchar_t *fgetws(wchar_t *, int, FILE *);
1419 extern wint_t fputwc(wchar_t, FILE *);
1420 extern int fputws(const wchar_t *, FILE *);
1421 extern int fwidew(FILE *, int);
1422 extern int fwprintf(FILE *, const wchar_t *, ...);
1423 extern int fwscanf(FILE *, const wchar_t *, ...);
1424 extern wint_t getwc(FILE *);
1425 extern wint_t getwchar(void);
1426 extern wint_t putwc(wchar_t, FILE *);
1427 extern wint_t putwchar(wchar_t);
1428 extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
1429 extern int swscanf(const wchar_t *, const wchar_t *, ...);
1430 extern wint_t ungetwc(wint_t, FILE *);
1431 extern int vfwprintf(FILE *, const wchar_t *, va_list);
1432 extern int vfwscanf(FILE *, const wchar_t *, va_list);
1433 extern int vswprintf(wchar_t *, size_t, const wchar_t *, va_list);
1434 extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
1435 extern int vwprintf(const wchar_t *, va_list);
1436 extern int vwscanf(const wchar_t *, va_list);
1437 extern size_t wcsftime(wchar_t *, size_t, const wchar_t *,
1438                       const struct tm *);
1439 extern int wprintf(const wchar_t *, ...);
1440 extern int wscanf(const wchar_t *, ...);

```

11.3.79 wctype.h

```

1441
1442 extern int iswblank(wint_t);
1443 extern wint_t towlower(wint_t);
1444 extern wint_t towupper(wint_t);
1445 extern wctrans_t wctrans(const char *);
1446 extern int iswalnum(wint_t);
1447 extern int iswalpha(wint_t);
1448 extern int iswcntrl(wint_t);
1449 extern int iswctype(wint_t, wctype_t);
1450 extern int iswdigit(wint_t);
1451 extern int iswgraph(wint_t);
1452 extern int iswlower(wint_t);
1453 extern int iswprint(wint_t);
1454 extern int iswpunct(wint_t);
1455 extern int iswspace(wint_t);
1456 extern int iswupper(wint_t);
1457 extern int iswxdigit(wint_t);
1458 extern wctype_t wctype(const char *);
1459 extern wint_t towctrans(wint_t, wctrans_t);

```

11.3.80 wordexp.h

```

1460
1461 extern int wordexp(const char *, wordexp_t *, int);
1462 extern void wordfree(wordexp_t *);

```

11.4 Interfaces for libm

1463 Table 11-24 defines the library name and shared object name for the libm library

1464 **Table 11-24 libm Definition**

Library:	libm
SONAME:	libm.so.6

1465

1466 The behavior of the interfaces in this library is specified by the following specifica-
1467 tions:

[ISOC99] ISO C (1999)
[LSB] This Specification
[SUSv2] SUSv2
1468 [SUSv3] ISO POSIX (2003)

11.4.1 Math

11.4.1.1 Interfaces for Math

1469

1470 An LSB conforming implementation shall provide the architecture specific functions
1471 for Math specified in Table 11-25, with the full mandatory functionality as described
1472 in the referenced underlying specification.

1473

Table 11-25 libm - Math Function Interfaces

<code>__finite(GLIBC_2.1)</code> [ISOC99]	<code>__finitef(GLIBC_2.1)</code> [ISOC99]	<code>__finitel(GLIBC_2.1)</code> [ISOC99]	<code>__fpclassify(GLIBC_2.1)</code> [LSB]
<code>__fpclassifyf(GLIBC_2.1)</code> [LSB]	<code>acos(GLIBC_2.0)</code> [SUSv3]	<code>acosf(GLIBC_2.0)</code> [SUSv3]	<code>acosh(GLIBC_2.0)</code> [SUSv3]
<code>acoshf(GLIBC_2.0)</code> [SUSv3]	<code>acoshl(GLIBC_2.0)</code> [SUSv3]	<code>acosl(GLIBC_2.0)</code> [SUSv3]	<code>asin(GLIBC_2.0)</code> [SUSv3]
<code>asinf(GLIBC_2.0)</code> [SUSv3]	<code>asinh(GLIBC_2.0)</code> [SUSv3]	<code>asinhf(GLIBC_2.0)</code> [SUSv3]	<code>asinhf(GLIBC_2.0)</code> [SUSv3]
<code>asinl(GLIBC_2.0)</code> [SUSv3]	<code>atan(GLIBC_2.0)</code> [SUSv3]	<code>atan2(GLIBC_2.0)</code> [SUSv3]	<code>atan2f(GLIBC_2.0)</code> [SUSv3]
<code>atan2l(GLIBC_2.0)</code> [SUSv3]	<code>atanf(GLIBC_2.0)</code> [SUSv3]	<code>atanh(GLIBC_2.0)</code> [SUSv3]	<code>atanhf(GLIBC_2.0)</code> [SUSv3]
<code>atanhl(GLIBC_2.0)</code> [SUSv3]	<code>atanl(GLIBC_2.0)</code> [SUSv3]	<code>cabs(GLIBC_2.1)</code> [SUSv3]	<code>cabsf(GLIBC_2.1)</code> [SUSv3]
<code>cabsl(GLIBC_2.1)</code> [SUSv3]	<code>cacos(GLIBC_2.1)</code> [SUSv3]	<code>cacosf(GLIBC_2.1)</code> [SUSv3]	<code>cacosh(GLIBC_2.1)</code> [SUSv3]
<code>cacoshf(GLIBC_2.1)</code> [SUSv3]	<code>cacoshl(GLIBC_2.1)</code> [SUSv3]	<code>cacosl(GLIBC_2.1)</code> [SUSv3]	<code>carg(GLIBC_2.1)</code> [SUSv3]
<code>cargf(GLIBC_2.1)</code> [SUSv3]	<code>cargl(GLIBC_2.1)</code> [SUSv3]	<code>casin(GLIBC_2.1)</code> [SUSv3]	<code>casinf(GLIBC_2.1)</code> [SUSv3]
<code>casinh(GLIBC_2.1)</code> [SUSv3]	<code>casinhf(GLIBC_2.1)</code> [SUSv3]	<code>casinhf(GLIBC_2.1)</code> [SUSv3]	<code>casinl(GLIBC_2.1)</code> [SUSv3]
<code>catan(GLIBC_2.1)</code> [SUSv3]	<code>catanf(GLIBC_2.1)</code> [SUSv3]	<code>catanh(GLIBC_2.1)</code> [SUSv3]	<code>catanhf(GLIBC_2.1)</code> [SUSv3]
<code>catanhl(GLIBC_2.1)</code> [SUSv3]	<code>catanl(GLIBC_2.1)</code> [SUSv3]	<code>cbrt(GLIBC_2.0)</code> [SUSv3]	<code>cbrtf(GLIBC_2.0)</code> [SUSv3]
<code>cbrtl(GLIBC_2.0)</code> [SUSv3]	<code>ccos(GLIBC_2.1)</code> [SUSv3]	<code>ccosf(GLIBC_2.1)</code> [SUSv3]	<code>ccosh(GLIBC_2.1)</code> [SUSv3]
<code>ccoshf(GLIBC_2.1)</code>	<code>ccoshl(GLIBC_2.1)</code>	<code>ccosl(GLIBC_2.1)</code>	<code>ceil(GLIBC_2.0)</code>

) [SUSv3]	[SUSv3]	[SUSv3]	[SUSv3]
ceilf(GLIBC_2.0) [SUSv3]	ceil(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) [SUSv3]	erfl(GLIBC_2.0) [SUSv3]	exp(GLIBC_2.0) [SUSv3]	exp2(GLIBC_2.1) [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]
finitf(GLIBC_2.0) [ISO99]	finitel(GLIBC_2.0) [ISO99]	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) [ISO99]	gammal(GLIBC_2.0) [ISO99]	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) [ISO99]	j0l(GLIBC_2.0) [ISO99]
j1(GLIBC_2.0) [SUSv3]	j1f(GLIBC_2.0) [ISO99]	j1l(GLIBC_2.0) [ISO99]	jn(GLIBC_2.0) [SUSv3]
jnf(GLIBC_2.0) [ISO99]	jnl(GLIBC_2.0) [ISO99]	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) [ISO99]	lgammaf(GLIBC_2.0) [SUSv3]
lgammaf_r(GLIBC_2.0) [ISO99]	lgammal(GLIBC_2.0) [SUSv3]	lgammal_r(GLIBC_2.0) [ISO99]	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) [ISO99]	modf(GLIBC_2.0) [SUSv3]
modff(GLIBC_2.0)	modfl(GLIBC_2.0)	nan(GLIBC_2.1)	nanf(GLIBC_2.1)

[SUSv3]	[SUSv3]	[SUSv3]	[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) [SUSv3]	tanl(GLIBC_2.0) [SUSv3]	tgamma(GLIBC_2.1) [SUSv3]	tgammalf(GLIBC_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]			

1475 An LSB conforming implementation shall provide the architecture specific data
 1476 interfaces for Math specified in Table 11-26, with the full mandatory functionality as
 1477 described in the referenced underlying specification.

1478 **Table 11-26 libm - Math Data Interfaces**

1479	signgam(GLIBC_2 .0) [SUSv3]			
------	--------------------------------	--	--	--

11.5 Data Definitions for libm

1480 This section defines global identifiers and their values that are associated with
 1481 interfaces contained in libm. These definitions are organized into groups that
 1482 correspond to system headers. This convention is used as a convenience for the
 1483 reader, and does not imply the existence of these headers, or their content. Where an
 1484 interface is defined as requiring a particular system header file all of the data
 1485 definitions for that system header file presented here shall be in effect.

1486 This section gives data definitions to promote binary application portability, not to
 1487 repeat source interface definitions available elsewhere. System providers and
 1488 application developers should use this ABI to supplement - not to replace - source
 1489 interface definition specifications.

1490 This specification uses the ISO C (1999) C Language as the reference programming
 1491 language, and data definitions are specified in ISO C format. The C language is used
 1492 here as a convenient notation. Using a C language description of these data objects
 1493 does not preclude their use by other programming languages.

11.5.1 complex.h

```

1494 extern double cabs(double complex);
1495 extern float cabsf(float complex);
1496 extern long double cabsl(long double complex);
1497 extern double complex cacos(double complex);
1498 extern float complex cacosf(float complex);
1499 extern double complex cacosh(double complex);
1500 extern float complex cacoshf(float complex);
1501 extern long double complex cacoshl(long double complex);
1502 extern long double complex cacosl(long double complex);
1503 extern double carg(double complex);
1504 extern float cargf(float complex);
1505 extern long double cargl(long double complex);
1506 extern double complex casin(double complex);
1507 extern float complex casinf(float complex);
1508 extern double complex casinh(double complex);
1509 extern float complex casinhf(float complex);
1510 extern long double complex casinhl(long double complex);
1511 extern long double complex casinl(long double complex);
1512 extern double complex catan(double complex);
1513 extern float complex catanf(float complex);
1514 extern double complex catanh(double complex);
1515 extern float complex catanhf(float complex);
1516 extern long double complex catanhl(long double complex);
1517 extern long double complex catanl(long double complex);
1518 extern double complex ccos(double complex);
1519 extern float complex ccosf(float complex);
1520 extern double complex ccosh(double complex);
1521 extern float complex ccoshf(float complex);
1522 extern long double complex ccoshl(long double complex);

```

```

1524     extern long double complex ccosl(long double complex);
1525     extern double complex cexp(double complex);
1526     extern float complex cexpf(float complex);
1527     extern long double complex cexpl(long double complex);
1528     extern double cimag(double complex);
1529     extern float cimagf(float complex);
1530     extern long double cimagl(long double complex);
1531     extern double complex clog(double complex);
1532     extern float complex clog10f(float complex);
1533     extern long double complex clog10l(long double complex);
1534     extern float complex clogf(float complex);
1535     extern long double complex clogl(long double complex);
1536     extern double complex conj(double complex);
1537     extern float complex conjf(float complex);
1538     extern long double complex conjl(long double complex);
1539     extern double complex cpow(double complex, double complex);
1540     extern float complex cpowf(float complex, float complex);
1541     extern long double complex cpowl(long double complex, long double
1542     complex);
1543     extern double complex cproj(double complex);
1544     extern float complex cprojf(float complex);
1545     extern long double complex cprojl(long double complex);
1546     extern double creal(double complex);
1547     extern float crealf(float complex);
1548     extern long double creall(long double complex);
1549     extern double complex csin(double complex);
1550     extern float complex csinf(float complex);
1551     extern double complex csinh(double complex);
1552     extern float complex csinhf(float complex);
1553     extern long double complex csinhl(long double complex);
1554     extern long double complex csinl(long double complex);
1555     extern double complex csqrt(double complex);
1556     extern float complex csqrtf(float complex);
1557     extern long double complex csqrtl(long double complex);
1558     extern double complex ctan(double complex);
1559     extern float complex ctanf(float complex);
1560     extern double complex ctanh(double complex);
1561     extern float complex ctanhf(float complex);
1562     extern long double complex ctanhl(long double complex);
1563     extern long double complex ctanl(long double complex);

```

11.5.2 fenv.h

```

1564
1565     #define FE_INEXACT      0x08
1566     #define FE_UNDERFLOW  0x10
1567     #define FE_OVERFLOW   0x20
1568     #define FE_DIVBYZERO  0x40
1569     #define FE_INVALID    0x80
1570
1571     #define FE_ALL_EXCEPT \
1572     (FE_INEXACT | FE_DIVBYZERO | FE_UNDERFLOW | FE_OVERFLOW |
1573     FE_INVALID)
1574
1575     #define FE_TONEAREST  0
1576     #define FE_TOWARDZERO 0x1
1577     #define FE_UPWARD     0x2
1578     #define FE_DOWNWARD   0x3
1579
1580     typedef unsigned int fexcept_t;
1581
1582     typedef struct {
1583         fexcept_t fpc;
1584         void *ieee_instruction_pointer;

```

```

1585     } fenv_t;
1586
1587     #define FE_DFL_ENV      ((__const fenv_t *) -1)
1588
1589     extern int feclearexcept(int);
1590     extern int fegetenv(fenv_t *);
1591     extern int fegetexceptflag(fexcept_t *, int);
1592     extern int fegetround(void);
1593     extern int feholdexcept(fenv_t *);
1594     extern int feraiseexcept(int);
1595     extern int fesetenv(const fenv_t *);
1596     extern int fesetexceptflag(const fexcept_t *, int);
1597     extern int fesetround(int);
1598     extern int fetestexcept(int);
1599     extern int feupdateenv(const fenv_t *);

```

11.5.3 math.h

```

1600
1601     #define fpclassify(x)  \
1602         (sizeof (x) == sizeof (float) ? __fpclassifyf (x) : __fpclassify
1603         (x) )
1604     #define signbit(x)    \
1605         (sizeof (x) == sizeof (float)? __signbitf (x): __signbit (x))
1606
1607     #define FP_ILOGB0      -2147483647
1608     #define FP_ILOGBNAN   2147483647
1609
1610     extern int __finite(double);
1611     extern int __finitef(float);
1612     extern int __finitel(long double);
1613     extern int __isinf(double);
1614     extern int __isinff(float);
1615     extern int __isinfl(long double);
1616     extern int __isnan(double);
1617     extern int __isnanf(float);
1618     extern int __isnanl(long double);
1619     extern int __signbit(double);
1620     extern int __signbitf(float);
1621     extern int __fpclassify(double);
1622     extern int __fpclassifyf(float);
1623     extern int __fpclassifyl(long double);
1624     extern int signgam(void);
1625     extern double copysign(double, double);
1626     extern int finite(double);
1627     extern double frexp(double, int *);
1628     extern double ldexp(double, int);
1629     extern double modf(double, double *);
1630     extern double acos(double);
1631     extern double acosh(double);
1632     extern double asinh(double);
1633     extern double atanh(double);
1634     extern double asin(double);
1635     extern double atan(double);
1636     extern double atan2(double, double);
1637     extern double cbrt(double);
1638     extern double ceil(double);
1639     extern double cos(double);
1640     extern double cosh(double);
1641     extern double erf(double);
1642     extern double erfc(double);
1643     extern double exp(double);
1644     extern double expm1(double);
1645     extern double fabs(double);

```

```

1646     extern double floor(double);
1647     extern double fmod(double, double);
1648     extern double gamma(double);
1649     extern double hypot(double, double);
1650     extern int ilogb(double);
1651     extern double j0(double);
1652     extern double j1(double);
1653     extern double jn(int, double);
1654     extern double lgamma(double);
1655     extern double log(double);
1656     extern double log10(double);
1657     extern double loglp(double);
1658     extern double logb(double);
1659     extern double nextafter(double, double);
1660     extern double pow(double, double);
1661     extern double remainder(double, double);
1662     extern double rint(double);
1663     extern double scalb(double, double);
1664     extern double sin(double);
1665     extern double sinh(double);
1666     extern double sqrt(double);
1667     extern double tan(double);
1668     extern double tanh(double);
1669     extern double y0(double);
1670     extern double y1(double);
1671     extern double yn(int, double);
1672     extern float copysignf(float, float);
1673     extern long double copysignl(long double, long double);
1674     extern int finitef(float);
1675     extern int finitel(long double);
1676     extern float frexpf(float, int *);
1677     extern long double frexpl(long double, int *);
1678     extern float ldexpf(float, int);
1679     extern long double ldexpl(long double, int);
1680     extern float modff(float, float *);
1681     extern long double modfl(long double, long double *);
1682     extern double scalbln(double, long int);
1683     extern float scalblnf(float, long int);
1684     extern long double scalblnl(long double, long int);
1685     extern double scalbn(double, int);
1686     extern float scalbnf(float, int);
1687     extern long double scalbnl(long double, int);
1688     extern float acosf(float);
1689     extern float acoshf(float);
1690     extern long double acoshl(long double);
1691     extern long double acosl(long double);
1692     extern float asinf(float);
1693     extern float asinhf(float);
1694     extern long double asinhl(long double);
1695     extern long double asinl(long double);
1696     extern float atan2f(float, float);
1697     extern long double atan2l(long double, long double);
1698     extern float atanf(float);
1699     extern float atanhf(float);
1700     extern long double atanhl(long double);
1701     extern long double atanl(long double);
1702     extern float cbrtf(float);
1703     extern long double cbrtl(long double);
1704     extern float ceilf(float);
1705     extern long double ceill(long double);
1706     extern float cosf(float);
1707     extern float coshf(float);
1708     extern long double coshl(long double);
1709     extern long double cosl(long double);

```

11 Libraries

```
1710     extern float dremf(float, float);
1711     extern long double dreml(long double, long double);
1712     extern float erfcf(float);
1713     extern long double erfcl(long double);
1714     extern float erff(float);
1715     extern long double erfl(long double);
1716     extern double exp2(double);
1717     extern float exp2f(float);
1718     extern long double exp2l(long double);
1719     extern float expf(float);
1720     extern long double expl(long double);
1721     extern float expmlf(float);
1722     extern long double expmll(long double);
1723     extern float fabsf(float);
1724     extern long double fabsl(long double);
1725     extern double fdim(double, double);
1726     extern float fdimf(float, float);
1727     extern long double fdiml(long double, long double);
1728     extern float floorf(float);
1729     extern long double floorl(long double);
1730     extern double fma(double, double, double);
1731     extern float fmaf(float, float, float);
1732     extern long double fmal(long double, long double, long double);
1733     extern double fmax(double, double);
1734     extern float fmaxf(float, float);
1735     extern long double fmaxl(long double, long double);
1736     extern double fmin(double, double);
1737     extern float fminf(float, float);
1738     extern long double fminl(long double, long double);
1739     extern float fmodf(float, float);
1740     extern long double fmodl(long double, long double);
1741     extern float gammaf(float);
1742     extern long double gammal(long double);
1743     extern float hypotf(float, float);
1744     extern long double hypotl(long double, long double);
1745     extern int ilogbf(float);
1746     extern int ilogbl(long double);
1747     extern float j0f(float);
1748     extern long double j0l(long double);
1749     extern float j1f(float);
1750     extern long double j1l(long double);
1751     extern float jnf(int, float);
1752     extern long double jnl(int, long double);
1753     extern double lgamma_r(double, int *);
1754     extern float lgammaf(float);
1755     extern float lgammaf_r(float, int *);
1756     extern long double lgammal(long double);
1757     extern long double lgammal_r(long double, int *);
1758     extern long long int llrint(double);
1759     extern long long int llrintf(float);
1760     extern long long int llrintl(long double);
1761     extern long long int llround(double);
1762     extern long long int llroundf(float);
1763     extern long long int llroundl(long double);
1764     extern float log10f(float);
1765     extern long double log10l(long double);
1766     extern float log1pf(float);
1767     extern long double log1pl(long double);
1768     extern double log2(double);
1769     extern float log2f(float);
1770     extern long double log2l(long double);
1771     extern float logbf(float);
1772     extern long double logbl(long double);
1773     extern float logf(float);
```

```

1774     extern long double logl(long double);
1775     extern long int lrint(double);
1776     extern long int lrintf(float);
1777     extern long int lrintl(long double);
1778     extern long int lround(double);
1779     extern long int lroundf(float);
1780     extern long int lroundl(long double);
1781     extern int matherr(struct exception *);
1782     extern double nan(const char *);
1783     extern float nanf(const char *);
1784     extern long double nanl(const char *);
1785     extern double nearbyint(double);
1786     extern float nearbyintf(float);
1787     extern long double nearbyintl(long double);
1788     extern float nextafterf(float, float);
1789     extern long double nextafterl(long double, long double);
1790     extern double nexttoward(double, long double);
1791     extern float nexttowardf(float, long double);
1792     extern long double nexttowardl(long double, long double);
1793     extern double powl0(double);
1794     extern float powl0f(float);
1795     extern long double powl0l(long double);
1796     extern float powf(float, float);
1797     extern long double powl(long double, long double);
1798     extern float remainderf(float, float);
1799     extern long double remainderl(long double, long double);
1800     extern double remquo(double, double, int *);
1801     extern float remquof(float, float, int *);
1802     extern long double remquol(long double, long double, int *);
1803     extern float rintf(float);
1804     extern long double rintl(long double);
1805     extern double round(double);
1806     extern float roundf(float);
1807     extern long double roundl(long double);
1808     extern float scalbf(float, float);
1809     extern long double scalbl(long double, long double);
1810     extern double significand(double);
1811     extern float significandf(float);
1812     extern long double significandl(long double);
1813     extern void sincos(double, double *, double *);
1814     extern void sincosf(float, float *, float *);
1815     extern void sincosl(long double, long double *, long double *);
1816     extern float sinf(float);
1817     extern float sinhf(float);
1818     extern long double sinhl(long double);
1819     extern long double sinl(long double);
1820     extern float sqrtf(float);
1821     extern long double sqrtl(long double);
1822     extern float tanf(float);
1823     extern float tanhf(float);
1824     extern long double tanhl(long double);
1825     extern long double tanl(long double);
1826     extern double tgamma(double);
1827     extern float tgammaf(float);
1828     extern long double tgamma1(long double);
1829     extern double trunc(double);
1830     extern float truncf(float);
1831     extern long double trunc1(long double);
1832     extern float y0f(float);
1833     extern long double y0l(long double);
1834     extern float y1f(float);
1835     extern long double y1l(long double);
1836     extern float ynf(int, float);
1837     extern long double ynl(int, long double);

```

```

1838     extern int __fpclassifyl(long double);
1839     extern int __fpclassifyl(long double);
1840     extern int __signbitl(long double);
1841     extern int __signbitl(long double);
1842     extern int __signbitl(long double);
1843     extern long double exp2l(long double);
1844     extern long double exp2l(long double);
    
```

11.6 Interfaces for libpthread

1845 Table 11-27 defines the library name and shared object name for the libpthread
 1846 library

1847 **Table 11-27 libpthread Definition**

Library:	libpthread
SONAME:	libpthread.so.0

1848
 1849 The behavior of the interfaces in this library is specified by the following specifica-
 1850 tions:

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

1851

11.6.1 Realtime Threads

11.6.1.1 Interfaces for Realtime Threads

1852

1853 An LSB conforming implementation shall provide the architecture specific functions
 1854 for Realtime Threads specified in Table 11-28, with the full mandatory functionality
 1855 as described in the referenced underlying specification.

1856 **Table 11-28 libpthread - Realtime Threads Function Interfaces**

pthread_attr_getinheritched(GLIBC_2.0) [SUSv3]	pthread_attr_getschedpolicy(GLIBC_2.0) [SUSv3]	pthread_attr_getschedparam(GLIBC_2.0) [SUSv3]	pthread_attr_setinheritched(GLIBC_2.0) [SUSv3]
pthread_attr_setschedpolicy(GLIBC_2.0) [SUSv3]	pthread_attr_setschedparam(GLIBC_2.0) [SUSv3]	pthread_getschedparam(GLIBC_2.0) [SUSv3]	pthread_setschedparam(GLIBC_2.0) [SUSv3]

1857

11.6.2 Advanced Realtime Threads

11.6.2.1 Interfaces for Advanced Realtime Threads

1858

1859 No external functions are defined for libpthread - Advanced Realtime Threads in
 1860 this part of the specification. See also the generic specification.

11.6.3 Posix Threads

11.6.3.1 Interfaces for Posix Threads

1861

1862 An LSB conforming implementation shall provide the architecture specific functions
 1863 for Posix Threads specified in Table 11-29, with the full mandatory functionality as
 1864 described in the referenced underlying specification.

Table 11-29 libpthread - Posix Threads Function Interfaces

_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_getshared(GLIBC_2.2) [SUSv3]
pthread_condattr_init(GLIBC_2.0) [SUSv3]	pthread_condattr_setshared(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) [SUSv3]	pthread_mutexattr_destroy(GLIBC_2.0) [SUSv3]	pthread_mutexattr_getshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_gettype(GLIBC_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_	pthread_rwlockat	pthread_rwlockat	pthread_rwlockat

1866	wrlock(GLIBC_2.1) [SUSv3]	tr_destroy(GLIBC_2.1) [SUSv3]	tr_getpshared(GLIBC_2.1) [SUSv3]	tr_init(GLIBC_2.1) [SUSv3]
	pthread_rwlockat tr_setpshared(GLIBC_2.1) [SUSv3]	pthread_self(GLIBC_2.0) [SUSv3]	pthread_setcancel state(GLIBC_2.0) [SUSv3]	pthread_setcancel type(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]		

11.6.4 Thread aware versions of libc interfaces

11.6.4.1 Interfaces for Thread aware versions of libc interfaces

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

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

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

11.7 Data Definitions for libpthread

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

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

This specification uses the 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.

11.7.1 pthread.h

```

1888
1889 extern void _pthread_cleanup_pop(struct _pthread_cleanup_buffer *,
1890 int);
1891 extern void _pthread_cleanup_push(struct _pthread_cleanup_buffer *,
1892 void (*__routine) (void *)
1893 , void *);
1894 extern int pthread_attr_destroy(pthread_attr_t *);
1895 extern int pthread_attr_getdetachstate(const typedef struct {
1896 int __detachstate;
1897 int __schedpolicy;
1898 struct sched_param
1899 __schedparam;
1900 int __inheritsched;
1901 int __scope;
1902 size_t __guardsize;
1903 int __stackaddr_set;
1904 void *__stackaddr;
1905 unsigned long int __stacksize;}
1906 pthread_attr_t *, int *);
1907 extern int pthread_attr_getinheritsched(const typedef struct {
1908 int __detachstate;
1909 int __schedpolicy;
1910 struct sched_param
1911 __schedparam;
1912 int __inheritsched;
1913 int __scope;
1914 size_t __guardsize;
1915 int __stackaddr_set;
1916 void *__stackaddr;
1917 unsigned long int
1918 __stacksize;}
1919 pthread_attr_t *, int *);
1920 extern int pthread_attr_getschedparam(const typedef struct {
1921 int __detachstate;
1922 int __schedpolicy;
1923 struct sched_param
1924 __schedparam;
1925 int __inheritsched;
1926 int __scope;
1927 size_t __guardsize;
1928 int __stackaddr_set;
1929 void *__stackaddr;
1930 unsigned long int __stacksize;}
1931 pthread_attr_t *, struct
1932 sched_param {
1933 int sched_priority;}
1934
1935 *);
1936 extern int pthread_attr_getschedpolicy(const typedef struct {
1937 int __detachstate;
1938 int __schedpolicy;
1939 struct sched_param
1940 __schedparam;
1941 int __inheritsched;
1942 int __scope;
1943 size_t __guardsize;
1944 int __stackaddr_set;
1945 void *__stackaddr;
1946 unsigned long int __stacksize;}
1947 pthread_attr_t *, int *);
1948 extern int pthread_attr_getscope(const typedef struct {
1949 int __detachstate;

```

```

1950             int __schedpolicy;
1951             struct sched_param __schedparam;
1952             int __inheritsched;
1953             int __scope;
1954             size_t __guardsize;
1955             int __stackaddr_set;
1956             void *__stackaddr;
1957             unsigned long int __stacksize;}
1958         pthread_attr_t *, int *);
1959     extern int pthread_attr_init(pthread_attr_t *);
1960     extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
1961     extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
1962     extern int pthread_attr_setschedparam(pthread_attr_t *, const struct
1963     sched_param {
1964                 int sched_priority;}
1965
1966                 *);
1967     extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
1968     extern int pthread_attr_setscope(pthread_attr_t *, int);
1969     extern int pthread_cancel(typedef unsigned long int pthread_t);
1970     extern int pthread_cond_broadcast(pthread_cond_t *);
1971     extern int pthread_cond_destroy(pthread_cond_t *);
1972     extern int pthread_cond_init(pthread_cond_t *, const typedef struct {
1973                 int __dummy;}
1974
1975                 pthread_condattr_t *);
1976     extern int pthread_cond_signal(pthread_cond_t *);
1977     extern int pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
1978     const struct timespec {
1979                 time_t tv_sec; long int tv_nsec;}
1980
1981                 *);
1982     extern int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
1983     extern int pthread_condattr_destroy(pthread_condattr_t *);
1984     extern int pthread_condattr_init(pthread_condattr_t *);
1985     extern int pthread_create(pthread_t *, const typedef struct {
1986                 int __detachstate;
1987                 int __schedpolicy;
1988                 struct sched_param __schedparam;
1989                 int __inheritsched;
1990                 int __scope;
1991                 size_t __guardsize;
1992                 int __stackaddr_set;
1993                 void *__stackaddr;
1994                 unsigned long int __stacksize;}
1995                 pthread_attr_t *,
1996                 void *(*__start_routine) (void *p1)
1997                 , void *);
1998     extern int pthread_detach(typedef unsigned long int pthread_t);
1999     extern int pthread_equal(typedef unsigned long int pthread_t,
2000                             typedef unsigned long int pthread_t);
2001     extern void pthread_exit(void *);
2002     extern int pthread_getschedparam(typedef unsigned long int pthread_t,
2003                                     int *, struct sched_param {
2004                                     int sched_priority;}
2005
2006                                     *);
2007     extern void *pthread_getspecific(typedef unsigned int pthread_key_t);
2008     extern int pthread_join(typedef unsigned long int pthread_t, void **);
2009     extern int pthread_key_create(pthread_key_t *, void (*destr_func) (void
2010     *)
2011     );
2012     extern int pthread_key_delete(typedef unsigned int pthread_key_t);
2013     extern int pthread_mutex_destroy(pthread_mutex_t *);

```

```

2014 extern int pthread_mutex_init(pthread_mutex_t *, const typedef struct
2015 {
2016     int __mutexkind;}
2017
2018     pthread_mutexattr_t *);
2019 extern int pthread_mutex_lock(pthread_mutex_t *);
2020 extern int pthread_mutex_trylock(pthread_mutex_t *);
2021 extern int pthread_mutex_unlock(pthread_mutex_t *);
2022 extern int pthread_mutexattr_destroy(pthread_mutexattr_t *);
2023 extern int pthread_mutexattr_init(pthread_mutexattr_t *);
2024 extern int pthread_once(pthread_once_t *, void (*init_routine) (void)
2025     );
2026 extern int pthread_rwlock_destroy(pthread_rwlock_t *);
2027 extern int pthread_rwlock_init(pthread_rwlock_t *,
2028     pthread_rwlockattr_t *);
2029 extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
2030 extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *);
2031 extern int pthread_rwlock_trywrlock(pthread_rwlock_t *);
2032 extern int pthread_rwlock_unlock(pthread_rwlock_t *);
2033 extern int pthread_rwlock_wrlock(pthread_rwlock_t *);
2034 extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
2035 extern int pthread_rwlockattr_getpshared(const typedef struct {
2036     int __lockkind; int
2037     __pshared;}
2038     pthread_rwlockattr_t *, int
2039     *);
2040 extern int pthread_rwlockattr_init(pthread_rwlockattr_t *);
2041 extern int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
2042 extern typedef unsigned long int pthread_t pthread_self(void);
2043 extern int pthread_setcancelstate(int, int *);
2044 extern int pthread_setcanceltype(int, int *);
2045 extern int pthread_setschedparam(typedef unsigned long int pthread_t,
2046     int, const struct sched_param {
2047     int sched_priority;}
2048
2049     *);
2050 extern int pthread_setspecific(typedef unsigned int pthread_key_t,
2051     const void *);
2052 extern void pthread_testcancel(void);
2053 extern int pthread_attr_getguardsize(const typedef struct {
2054     int __detachstate;
2055     int __schedpolicy;
2056     struct sched_param __schedparam;
2057     int __inheritsched;
2058     int __scope;
2059     size_t __guardsize;
2060     int __stackaddr_set;
2061     void *__stackaddr;
2062     unsigned long int __stacksize;}
2063     pthread_attr_t *, size_t *);
2064 extern int pthread_attr_setguardsize(pthread_attr_t *,
2065     typedef unsigned long int
2066     size_t);
2067 extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
2068 extern int pthread_attr_getstackaddr(const typedef struct {
2069     int __detachstate;
2070     int __schedpolicy;
2071     struct sched_param __schedparam;
2072     int __inheritsched;
2073     int __scope;
2074     size_t __guardsize;
2075     int __stackaddr_set;
2076     void *__stackaddr;
2077     unsigned long int __stacksize;}

```

```

2078                                     pthread_attr_t *, void **);
2079 extern int pthread_attr_setstacksize(pthread_attr_t *,
2080                                     typedef unsigned long int
2081 size_t);
2082 extern int pthread_attr_getstacksize(const typedef struct {
2083                                     int __detachstate;
2084                                     int __schedpolicy;
2085                                     struct sched_param __schedparam;
2086                                     int __inheritsched;
2087                                     int __scope;
2088                                     size_t __guardsize;
2089                                     int __stackaddr_set;
2090                                     void *__stackaddr;
2091                                     unsigned long int __stacksize;}
2092                                     pthread_attr_t *, size_t *);
2093 extern int pthread_mutexattr_gettype(const typedef struct {
2094                                     int __mutexkind;}
2095                                     pthread_mutexattr_t *, int *);
2096 extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
2097 extern int pthread_getconcurrency(void);
2098 extern int pthread_setconcurrency(int);
2099 extern int pthread_attr_getstack(const typedef struct {
2100                                     int __detachstate;
2101                                     int __schedpolicy;
2102                                     struct sched_param __schedparam;
2103                                     int __inheritsched;
2104                                     int __scope;
2105                                     size_t __guardsize;
2106                                     int __stackaddr_set;
2107                                     void *__stackaddr;
2108                                     unsigned long int __stacksize;}
2109                                     pthread_attr_t *, void **, size_t *);
2110 extern int pthread_attr_setstack(pthread_attr_t *, void *,
2111                                     typedef unsigned long int size_t);
2112 extern int pthread_condattr_getpshared(const typedef struct {
2113                                     int __dummy;}
2114                                     pthread_condattr_t *, int *);
2115 extern int pthread_condattr_setpshared(pthread_condattr_t *, int);
2116 extern int pthread_mutexattr_getpshared(const typedef struct {
2117                                     int __mutexkind;}
2118                                     pthread_mutexattr_t *, int *);
2119 extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
2120 extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *, const struct
2121 timespec {
2122                                     time_t tv_sec; long int
2123 tv_nsec; }
2124
2125                                     *);
2126 extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *, const struct
2127 timespec {
2128                                     time_t tv_sec; long int
2129 tv_nsec; }
2130
2131                                     *);
2132 extern int __register_atfork(void (*prepare) (void)
2133                             , void (*parent) (void)
2134                             , void (*child) (void)
2135                             , void *);
2136 extern int pthread_setschedprio(typedef unsigned long int pthread_t,
2137 int);

```

11.7.2 semaphore.h

2138

```

2139 extern int sem_close(sem_t *);
2140 extern int sem_destroy(sem_t *);
2141 extern int sem_getvalue(sem_t *, int *);
2142 extern int sem_init(sem_t *, int, unsigned int);
2143 extern sem_t *sem_open(const char *, int, ...);
2144 extern int sem_post(sem_t *);
2145 extern int sem_trywait(sem_t *);
2146 extern int sem_unlink(const char *);
2147 extern int sem_wait(sem_t *);
2148 extern int sem_timedwait(sem_t *, const struct timespec *);

```

11.8 Interfaces for libgcc_s

2149 Table 11-31 defines the library name and shared object name for the libgcc_s library

2150 **Table 11-31 libgcc_s Definition**

Library:	libgcc_s
SONAME:	libgcc_s.so.1

2152 The behavior of the interfaces in this library is specified by the following specifica-
2153 tions:

2154 [LSB] This Specification

11.8.1 Unwind Library

11.8.1.1 Interfaces for Unwind Library

2155 An LSB conforming implementation shall provide the architecture specific functions
2156 for Unwind Library specified in Table 11-32, with the full mandatory functionality as
2157 described in the referenced underlying specification.
2158

2159 **Table 11-32 libgcc_s - Unwind Library Function Interfaces**

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

2160

11.9 Data Definitions for libgcc_s

2161 This section defines global identifiers and their values that are associated with
2162 interfaces contained in libgcc_s. These definitions are organized into groups that

2163 correspond to system headers. This convention is used as a convenience for the
 2164 reader, and does not imply the existence of these headers, or their content. Where an
 2165 interface is defined as requiring a particular system header file all of the data
 2166 definitions for that system header file presented here shall be in effect.

2167 This section gives data definitions to promote binary application portability, not to
 2168 repeat source interface definitions available elsewhere. System providers and
 2169 application developers should use this ABI to supplement - not to replace - source
 2170 interface definition specifications.

2171 This specification uses the ISO C (1999) C Language as the reference programming
 2172 language, and data definitions are specified in ISO C format. The C language is used
 2173 here as a convenient notation. Using a C language description of these data objects
 2174 does not preclude their use by other programming languages.

11.9.1 unwind.h

```

2175 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2176 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2177 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2178 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2179                                         _Unwind_Stop_Fn, void *);
2180
2181 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2182 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2183 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2184 _Unwind_Context
2185                                         *);
2186 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2187 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2188 _Unwind_Exception
2189                                         *);
2190 extern void _Unwind_Resume(struct _Unwind_Exception *);
2191 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2192 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2193 extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2194 extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2195 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2196                                         _Unwind_Stop_Fn, void *);
2197 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2198 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2199 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2200 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2201 _Unwind_Context
2202                                         *);
2203 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2204 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2205 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2206 _Unwind_Exception
2207                                         *);
2208 extern void _Unwind_Resume(struct _Unwind_Exception *);
2209 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2210 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2211 extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2212                                         _Unwind_Stop_Fn, void *);
2213 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2214 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2215 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2216 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2217 _Unwind_Context
2218                                         *);
2219 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);

```



```

2220     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2221     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2222     _Unwind_Exception
2223     *);
2224     extern void _Unwind_Resume(struct _Unwind_Exception *);
2225     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2226     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2227     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2228     extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2229     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2230     _Unwind_Stop_Fn, void *);
2231     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2232     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2233     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2234     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2235     _Unwind_Context
2236     *);
2237     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2238     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2239     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2240     _Unwind_Exception
2241     *);
2242     extern void _Unwind_Resume(struct _Unwind_Exception *);
2243     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2244     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2245     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2246     extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2247     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2248     _Unwind_Stop_Fn, void *);
2249     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2250     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2251     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2252     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2253     _Unwind_Context
2254     *);
2255     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2256     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2257     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2258     _Unwind_Exception
2259     *);
2260     extern void _Unwind_Resume(struct _Unwind_Exception *);
2261     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2262     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2263     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2264     extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2265     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2266     _Unwind_Stop_Fn, void *);
2267     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2268     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2269     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2270     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2271     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2272     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2273     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2274     _Unwind_Exception
2275     *);
2276     extern void _Unwind_Resume(struct _Unwind_Exception *);
2277     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2278     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2279     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2280     extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2281     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2282     _Unwind_Stop_Fn, void *);
2283     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);

```

```

2284     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2285     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2286     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2287     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2288     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2289     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2290     _Unwind_Exception
2291                                     *);
2292     extern void _Unwind_Resume(struct _Unwind_Exception *);
2293     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2294     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2295     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2296     *);
2297     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2298     *);
2299     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2300     *);
2301     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2302     *);
2303     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2304     *);
2305     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2306     *);
2307     extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2308     *);
2309     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2310     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2311     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2312     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2313     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2314     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2315     extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2316     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2317     _Unwind_Exception *);
2318     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2319     _Unwind_Exception *);
2320     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2321     _Unwind_Exception *);
2322     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2323     _Unwind_Exception *);
2324     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2325     _Unwind_Exception *);
2326     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2327     _Unwind_Exception *);
2328     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2329     _Unwind_Exception *);
2330     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2331     _Unwind_Exception *);
2332     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2333     _Unwind_Exception *);
2334     extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2335     _Unwind_Exception *);
2336     extern void *_Unwind_FindEnclosingFunction(void *);
2337     extern void *_Unwind_FindEnclosingFunction(void *);
2338     extern void *_Unwind_FindEnclosingFunction(void *);
2339     extern void *_Unwind_FindEnclosingFunction(void *);
2340     extern void *_Unwind_FindEnclosingFunction(void *);
2341     extern void *_Unwind_FindEnclosingFunction(void *);
2342     extern void *_Unwind_FindEnclosingFunction(void *);
2343     extern void *_Unwind_FindEnclosingFunction(void *);
2344     extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);

```

11.10 Interface Definitions for libgcc_s

2345 The interfaces defined on the following pages are included in libgcc_s and are
 2346 defined by this specification. Unless otherwise noted, these interfaces shall be
 2347 included in the source standard.

2348 Other interfaces listed in Section 11.8 shall behave as described in the referenced
 2349 base document.

_Unwind_DeleteException

Name

2350 _Unwind_DeleteException – private C++ error handling method

Synopsis

2351 void _Unwind_DeleteException(struct _Unwind_Exception * *object*);

Description

2352 _Unwind_DeleteException() deletes the given exception *object*. If a given
 2353 runtime resumes normal execution after catching a foreign exception, it will not
 2354 know how to delete that exception. Such an exception shall be deleted by calling
 2355 _Unwind_DeleteException(). This is a convenience function that calls the function
 2356 pointed to by the *exception_cleanup* field of the exception header.

_Unwind_Find_FDE

Name

2357 _Unwind_Find_FDE – private C++ error handling method

Synopsis

2358 fde * _Unwind_Find_FDE(void * *pc*, struct dwarf_eh_bases * *bases*);

Description

2359 _Unwind_Find_FDE() looks for the object containing *pc*, then inserts into *bases*.

_Unwind_ForcedUnwind

Name

2360 `_Unwind_ForcedUnwind` — private C++ error handling method

Synopsis

2361 `_Unwind_Reason_Code _Unwind_ForcedUnwind(struct _Unwind_Exception *`
2362 `object, _Unwind_Stop_Fn stop, void * stop_parameter);`

Description

2363 `_Unwind_ForcedUnwind()` raises an exception for forced unwinding, passing along
2364 the given exception *object*, which should have its *exception_class* and
2365 *exception_cleanup* fields set. The exception *object* has been allocated by the
2366 language-specific runtime, and has a language-specific format, except that it shall
2367 contain an `_Unwind_Exception` struct.

2368 Forced unwinding is a single-phase process. *stop* and *stop_parameter* control the
2369 termination of the unwind process instead of the usual personality routine query.
2370 *stop* is called for each unwind frame, with the parameteres described for the usual
2371 personality routine below, plus an additional *stop_parameter*.

Return Value

2372 When *stop* identifies the destination frame, it transfers control to the user code as
2373 appropriate without returning, normally after calling `_Unwind_DeleteException()`.
2374 If not, then it should return an `_Unwind_Reason_Code` value.

2375 If *stop* returns any reason code other than `_URC_NO_REASON`, then the stack state is
2376 indeterminate from the point of view of the caller of `_Unwind_ForcedUnwind()`.
2377 Rather than attempt to return, therefore, the unwind library should use the
2378 *exception_cleanup* entry in the exception, and then call `abort()`.

2379 `_URC_NO_REASON`

2380 This is not the destination from. The unwind runtime will call frame's
2381 personality routine with the `_UA_FORCE_UNWIND` and `_UA_CLEANUP_PHASE` flag
2382 set in *actions*, and then unwind to the next frame and call the `stop()` function
2383 again.

2384 `_URC_END_OF_STACK`

2385 In order to allow `_Unwind_ForcedUnwind()` to perform special processing
2386 when it reaches the end of the stack, the unwind runtime will call it after the last
2387 frame is rejected, with a NULL stack pointer in the context, and the `stop()`
2388 function shall catch this condition. It may return this code if it cannot handle
2389 end-of-stack.

2390 `_URC_FATAL_PHASE2_ERROR`

2391 The `stop()` function may return this code for other fatal conditions like stack
2392 corruption.

_Unwind_GetDataRelBase

Name

2393 `_Unwind_GetDataRelBase` – private IA64 C++ error handling method

Synopsis

2394 `_Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context * context);`

Description

2395 `_Unwind_GetDataRelBase()` returns the global pointer in register one for *context*.

_Unwind_GetGR

Name

2396 `_Unwind_GetGR` – private C++ error handling method

Synopsis

2397 `_Unwind_Word _Unwind_GetGR(struct _Unwind_Context * context, int index);`

Description

2398 `_Unwind_GetGR()` returns data at *index* found in *context*. The register is identified
2399 by its index: 0 to 31 are for the fixed registers, and 32 to 127 are for the stacked
2400 registers.

2401 During the two phases of unwinding, only GR1 has a guaranteed value, which is the
2402 global pointer of the frame referenced by the unwind *context*. If the register has its
2403 NAT bit set, the behavior is unspecified.

_Unwind_GetIP

Name

2404 `_Unwind_GetIP` – private C++ error handling method

Synopsis

2405 `_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);`

Description

2406 `_Unwind_GetIP()` returns the instruction pointer value for the routine identified by
2407 the unwind *context*.

_Unwind_GetLanguageSpecificData**Name**

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

Synopsis

2409 `_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *
2410 context, uint value);`

Description

2411 `_Unwind_GetLanguageSpecificData()` returns the address of the language specific
2412 data area for the current stack frame.

_Unwind_GetRegionStart**Name**

2413 `_Unwind_GetRegionStart` – private C++ error handling method

Synopsis

2414 `_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);`

Description

2415 `_Unwind_GetRegionStart()` routine returns the address (i.e., 0) of the beginning of
2416 the procedure or code fragment described by the current unwind descriptor block.

_Unwind_GetTextRelBase**Name**

2417 `_Unwind_GetTextRelBase` – private IA64 C++ error handling method

Synopsis

2418 `_Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context * context);`

Description

2419 `_Unwind_GetTextRelBase()` calls the abort method, then returns.

_Unwind_RaiseException

Name

2420 `_Unwind_RaiseException` – private C++ error handling method

Synopsis

2421 `_Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *
2422 object);`

Description

2423 `_Unwind_RaiseException()` raises an exception, passing along the given exception
2424 *object*, which should have its *exception_class* and *exception_cleanup* fields set.
2425 The exception object has been allocated by the language-specific runtime, and has a
2426 language-specific format, exception that it shall contain an `_Unwind_Exception`.

Return Value

2427 `_Unwind_RaiseException()` does not return unless an error condition is found. If
2428 an error condition occurs, an `_Unwind_Reason_Code` is returned:

2429 `_URC_END_OF_STACK`

2430 The unwinder encountered the end of the stack during phase one without
2431 finding a handler. The unwind runtime will not have modified the stack. The
2432 C++ runtime will normally call `uncaught_exception()` in this case.

2433 `_URC_FATAL_PHASE1_ERROR`

2434 The unwinder encountered an unexpected error during phase one, because of
2435 something like stack corruption. The unwind runtime will not have modified
2436 the stack. The C++ runtime will normally call `terminate()` in this case.

2437 `_URC_FATAL_PHASE2_ERROR`

2438 The unwinder encountered an unexpected error during phase two. This is
2439 usually a *throw*, which will call `terminate()`.

_Unwind_Resume

Name

2440 `_Unwind_Resume` – private C++ error handling method

Synopsis

2441 `void _Unwind_Resume(struct _Unwind_Exception * object);`

Description

2442 `_Unwind_Resume()` resumes propagation of an existing exception *object*. A call to
2443 this routine is inserted as the end of a landing pad that performs cleanup, but does
2444 not resume normal execution. It causes unwinding to proceed further.

_Unwind_SetGR

Name

2445 `_Unwind_SetGR` – private C++ error handling method

Synopsis

2446 `void _Unwind_SetGR(struct _Unwind_Context * context, int index, uint value);`

Description

2447 `_Unwind_SetGR()` sets the *value* of the register *indexed* for the routine identified by
2448 the unwind *context*.

_Unwind_SetIP

Name

2449 `_Unwind_SetIP` – private C++ error handling method

Synopsis

2450 `void _Unwind_SetIP(struct _Unwind_Context * context, uint value);`

Description

2451 `_Unwind_SetIP()` sets the *value* of the instruction pointer for the routine identified
2452 by the unwind *context*

11.11 Interfaces for libdl

2453 Table 11-33 defines the library name and shared object name for the libdl library

2454 **Table 11-33 libdl Definition**

Library:	libdl
SONAME:	libdl.so.2

2455

2456 The behavior of the interfaces in this library is specified by the following specifica-
2457 tions:

[LSB] This Specification

[SUSv3] ISO POSIX (2003)

2458

11.11.1 Dynamic Loader

11.11.1.1 Interfaces for Dynamic Loader

2459

2460 An LSB conforming implementation shall provide the architecture specific functions
2461 for Dynamic Loader specified in Table 11-34, with the full mandatory functionality
2462 as described in the referenced underlying specification.

2463 **Table 11-34 libdl - Dynamic Loader Function Interfaces**

<code>dladdr(GLIBC_2.0) [LSB]</code>	<code>dlclose(GLIBC_2.0) [SUSv3]</code>	<code>dLError(GLIBC_2. 0) [SUSv3]</code>	<code>dlopen(GLIBC_2. 1) [LSB]</code>
--	---	---	--

2464

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

11.12 Data Definitions for libdl

2465

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.

2466

2467

2468

2469

2470

2471

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.

2472

2473

2474

2475

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

2476

2477

2478

11.12.1 dlfcn.h

2479

2480

```
extern int dladdr(const void *, Dl_info *);
```

2481

```
extern int dlclose(void *);
```

2482

```
extern char *dlerror(void);
```

2483

```
extern void *dlopen(char *, int);
```

2484

```
extern void *dlsym(void *, char *);
```

11.13 Interfaces for libcrypt

2485

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

2486

Table 11-35 libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

2487

2488

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

2489

2490

[SUSv3] ISO POSIX (2003)

11.13.1 Encryption

2491

11.13.1.1 Interfaces for Encryption

2492

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.

2493

2494

2495

Table 11-36 libcrypt - Encryption Function Interfaces

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

2496

IV Utility Libraries

12 Libraries

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

12.1 Interfaces for libz

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

6 **Table 12-1 libz Definition**

Library:	libz
SONAME:	libz.so.1

7

12.1.1 Compression Library

8 12.1.1.1 Interfaces for Compression Library

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

12.2 Data Definitions for libz

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

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

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

12.2.1 zlib.h

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

```

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

```

12.3 Interfaces for libncurses

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

72 **Table 12-2 libncurses Definition**

73 Library:	libncurses
SONAME:	libncurses.so.5

12.3.1 Curses

74 12.3.1.1 Interfaces for Curses

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

12.4 Data Definitions for libncurses

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

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

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

12.4.1 curses.h

```

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

```

```

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

```

```

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

```

```

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

```



```

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

```

12.4.2 term.h

```

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

```

395 extern TERMINAL *cur_term(void);

12.5 Interfaces for libutil

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

397 **Table 12-3 libutil Definition**

Library:	libutil
SONAME:	libutil.so.1

398

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

401 [LSB] This Specification

12.5.1 Utility Functions

12.5.1.1 Interfaces for Utility Functions

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

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

forkpty(GLIBC_2.0) [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]		

407

V Package Format and Installation

13 Software Installation

13.1 Package Dependencies

1 The LSB runtime environment shall provide the following dependencies.

2 lsb-core-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 [LSB]

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

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

4

Annex B GNU Free Documentation License (Informative)

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

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

B.1 PREAMBLE

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

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

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

B.2 APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

B.3 VERBATIM COPYING

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

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

B.4 COPYING IN QUANTITY

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

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

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

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

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

B.5 MODIFICATIONS

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

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

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

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

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

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

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

B.6 COMBINING DOCUMENTS

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

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

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

B.7 COLLECTIONS OF DOCUMENTS

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

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

B.8 AGGREGATION WITH INDEPENDENT WORKS

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

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

B.9 TRANSLATION

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

B.10 TERMINATION

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

B.11 FUTURE REVISIONS OF THIS LICENSE

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

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

B.12 How to use this License for your documents

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

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

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

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