

Linux Standard Base Core Specification **for IA64 3.e1**

Linux Standard Base Core Specification for IA64 3.01

Copyright © 2004, 2005 Free Standards Group

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

Portions of the text are copyrighted by the following parties:

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

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

Linux is a trademark of Linus Torvalds.

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

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

AMD is a trademark of Advanced Micro Devices, Inc.

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

PowerPC and PowerPC Architecture are trademarks of the IBM Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Contents

| | |
|---|------------|
| Foreword | vi |
| Introduction | vii |
| I Introductory Elements | 8 |
| 1 Scope..... | 9 |
| 1.1 General..... | 9 |
| 1.2 Module Specific Scope..... | 9 |
| 2 Normative References..... | 10 |
| 3 Requirements 2.1 Normative References..... | 10 |
| 3.1 Relevant Libraries 2.2 Informative References/Bibliography | 13 |
| 3.2 LSB Implementation Conformance 3 Requirements..... | 16 |
| 3.3 LSB Application Conformance 3.1 Relevant Libraries..... | 16 |
| 4 Definitions 3.2 LSB Implementation Conformance..... | 16 |
| 5 Terminology 3.3 LSB Application Conformance | 17 |
| 6 Documentation Conventions 4 Definitions..... | 19 |
| II Executable and Linking Format (ELF) 5 Terminology..... | 20 |
| 7 Introduction 6 Documentation Conventions | 22 |
| 8 Low Level System Information II Executable and Linking Format (ELF) | 23 |
| 8.1 Machine Interface 7 Introduction | 24 |
| 8.2 Function Calling Sequence 8 Low Level System Information..... | 25 |
| 8.3 Operating System 1 Machine Interface | 25 |
| 8.4 Process Initialization 8.2 Function Calling Sequence | 29 |
| 8.5 Coding Examples 8.3 Operating System Interface | 30 |
| 8.6 C Stack Frame 8.4 Process Initialization..... | 31 |
| 8.7 Debug Information 8.5 Coding Examples | 33 |
| 9 Object Format 8.6 C Stack Frame | 34 |
| 9.1 Introduction 8.7 Debug Information | 35 |
| 9.2 ELF Header 9 Object Format..... | 36 |
| 9.3 Sections 9.1 Introduction..... | 36 |
| 9.4 Symbol Table 9.2 ELF Header | 36 |
| 9.5 Relocation 9.3 Sections..... | 37 |
| 10 Program Loading and Dynamic Linking 9.4 Symbol Table | 39 |
| 10.1 Introduction 9.5 Relocation..... | 39 |
| 10.2 Program Header Loading and Dynamic Linking | 40 |
| 10.3 Program Loading 10.1 Introduction | 40 |
| 10.4 Dynamic Linking 10.2 Program Header | 40 |
| III Base Libraries 10.3 Program Loading..... | 40 |
| 11 Libraries 10.4 Dynamic Linking | 40 |
| 11.1 Program Interpreter/Dynamic Linker III Base Libraries | 42 |
| 11.2 Interfaces for libe 11 Libraries | 43 |
| 11.3 Data Definitions for libe 11.1 Program Interpreter/Dynamic Linker..... | 43 |
| 11.4 Interfaces for libm libc | 43 |
| 11.5 Data Definitions for libm libc | 70 |
| 11.6 Interfaces for libpthread libm..... | 100 |
| 11.7 Interfaces for libgcc_s 11.5 Data Definitions for libm..... | 108 |
| 11.8 Interface Definitions for libgcc_s 11.6 Interfaces for libpthread | 114 |
| 11.9 Interfaces for libdl 11.7 Data Definitions for libpthread | 119 |
| 11.10 Interfaces for libcrypt libgcc_s..... | 123 |
| IV Utility Libraries 11.9 Data Definitions for libgcc_s..... | 124 |
| 12 Libraries 11.10 Interface Definitions for libgcc_s | 128 |

| | |
|--|------------|
| 12.11.11 Interfaces for libzlibdl..... | 132 |
| 12.2 Interfaces for libncurses 11.12 Data Definitions for libdl..... | 133 |
| 12.11.13 Interfaces for libutilibcrypt | 133 |
| V Package Format and InstallationIV Utility Libraries | 135 |
| 13 Software Installation12 Libraries | 136 |
| 13.1 Package Dependencies12.1 Interfaces for libz..... | 136 |
| 13.2 Package Architecture Considerations12.2 Data Definitions for libz..... | 136 |
| A Alphabetical Listing of Interfaces 12.3 Interfaces for libncurses..... | 137 |
| A.1 libgcc_s12.4 Data Definitions for libncurses..... | 137 |
| A.2 libm12.5 Interfaces for libutil..... | 143 |
| B GNU Free Documentation LicenseV Package Format and Installation | 144 |
| B.1 PREAMBLE13 Software Installation | 145 |
| B.2 APPLICABILITY AND DEFINITIONS13.1 Package Dependencies..... | 145 |
| B.3 VERBATIM COPYING13.2 Package Architecture Considerations | 145 |
| B.4 COPYING IN QUANTITY A Alphabetical Listing of Interfaces | 146 |
| B.5 MODIFICATIONS A.1 libgcc_s..... | 146 |
| B.6 COMBINING DOCUMENTS A.2 libm..... | 146 |
| B.7 COLLECTIONS OF DOCUMENTS B GNU Free Documentation License | |
| (Informative)..... | 147 |
| B.8 AGGREGATION WITH INDEPENDENT WORKS B.1 PREAMBLE..... | 147 |
| B.9 TRANSLATION B.2 APPLICABILITY AND DEFINITIONS | 147 |
| B.10 TERMINATION B.3 VERBATIM COPYING | 148 |
| B.11 FUTURE REVISIONS OF THIS LICENSE B.4 COPYING IN QUANTITY..... | 148 |
| B.12 How to use this License for your documents B.5 MODIFICATIONS..... | 149 |
| B.6 COMBINING DOCUMENTS..... | 150 |
| B.7 COLLECTIONS OF DOCUMENTS..... | 151 |
| B.8 AGGREGATION WITH INDEPENDENT WORKS..... | 151 |
| B.9 TRANSLATION | 151 |
| B.10 TERMINATION | 151 |
| B.11 FUTURE REVISIONS OF THIS LICENSE | 152 |
| B.12 How to use this License for your documents..... | 152 |

List of Figures

| | |
|---|----|
| 8-1 Structure Smaller Than A Word | 27 |
| 8-2 No Padding..... | 27 |
| 8-3 Internal and Tail Padding..... | 28 |
| 8-4 Bit-Field Ranges | 29 |

Foreword

1 | This is version 3.01 of the Linux Standard Base Core Specification for IA64. This
2 | specification is part of a family of specifications under the general title "Linux
3 | Standard Base". Developers of applications or implementations interested in using
4 | the LSB trademark should see the Free Standards Group Certification Policy for
5 | details.

Introduction

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

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

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

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

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

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

I Introductory Elements

1 Scope

1.1 General

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

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

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

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

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

1.2 Module Specific Scope

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

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

2 Normative References

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

2 References

2.1 Normative References

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

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

Table 2-1 Normative References

| Name | Title | URL |
|---|---|---|
| DWARF Debugging Information Format, Revision 2.0.0 | DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993) | http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf |
| DWARF Debugging Information Format, Revision 3.0.0 (Draft) | DWARF Debugging Information Format, Revision 3.0.0 (Draft) | http://refspecs.freestandards.org/dwarf/ |
| 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/ |
| Intel® Itanium™ Processor-specific Application Binary Interface | Intel® Itanium™ Processor-specific Application Binary Interface | http://refspecs.freestandards.org/elf/IA64-SysV-psABI.pdf |
| 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) | http://www.unix.org/version3/ |

| Name | Title | URL |
|--|--|---|
| | -- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004 | |
| ISO/IEC TR14652 | ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions | |
| Itanium™ Architecture Software Developer's Manual Volume 1 | Itanium™ Architecture Software Developer's Manual Volume 1: Application Architecture | http://refspecs.freestandards.org/IA64-softdevman-vol1.pdf |
| Itanium™ Architecture Software Developer's Manual Volume 2 | Itanium™ Architecture Software Developer's Manual Volume 2: System Architecture | http://refspecs.freestandards.org/IA64-softdevman-vol2.pdf |
| Itanium™ Architecture Software Developer's Manual Volume 3 | Itanium™ Architecture Software Developer's Manual Volume 3: Instruction Set Reference | http://refspecs.freestandards.org/IA64-softdevman-vol3.pdf |
| Itanium™ Architecture Software Developer's Manual Volume 4 | IA-64 Processor Reference: Intel® Itanium™ Processor Reference Manual for Software Development | http://refspecs.freestandards.org/IA64-softdevman-vol4.pdf |
| Itanium™ Software Conventions and Runtime Guide | Itanium™ Software Conventions & Runtime Architecture Guide, September 2000 | http://refspecs.freestandards.org/IA64conventions.pdf |

2 *Normative* References

| Name | Title | URL |
|--|--|---|
| 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 |
| Large File Support | Large File Support | http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html |
| Li18nux Globalization Specification | LI18NUNIX 2000 Globalization Specification, Version 1.0 with Amendment 4 | http://www.li18nux.org/docs/html/LI18NUNIX-2000-amd4.htm |
| Linux Allocated Device Registry | LINUX ALLOCATED DEVICES | http://www.lanana.org/docs/device-list/devices.txt |
| PAM | Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar & R.Schemers (SunSoft) | http://www.opengroup.org/tech/rfc/mirror/rfc/rfc86.0.txt |
| RFC 1321: The MD5 Message Digest Algorithm | IETF RFC 1321: The MD5 Message Digest Algorithm | http://www.ietf.org/rfc/rfc1321.txt |
| RFC 1833: Binding Protocols for ONC RPC Version 2 | IETF RFC 1833: Binding Protocols for ONC RPC Version 2 | http://www.ietf.org/rfc/rfc1833.txt |
| RFC 1950: ZLIB Compressed Data Format Specification | IETF RFC 1950: ZLIB Compressed Data Format Specification | http://www.ietf.org/rfc/rfc1950.txt |
| RFC 1951: DEFLATE Compressed Data Format Specification | IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3 | http://www.ietf.org/rfc/rfc1951.txt |
| RFC 1952: GZIP File Format Specification | IETF RFC 1952: GZIP file format specification version 4.3 | http://www.ietf.org/rfc/rfc1952.txt |
| RFC 2440: OpenPGP Message Format | IETF RFC 2440: OpenPGP Message Format | http://www.ietf.org/rfc/rfc2440.txt |
| RFC 2821: Simple Mail | IETF RFC 2821: Simple | http://www.ietf.org/rfc |

| Name | Title | URL |
|---|---|---|
| Transfer Protocol | Mail Transfer Protocol | /rfc2821.txt |
| RFC 2822:Internet Message Format | IETF RFC 2822: Internet Message Format | http://www.ietf.org/rfc/rfc2822.txt |
| RFC 791:Internet Protocol | IETF RFC 791: Internet Protocol Specification | http://www.ietf.org/rfc/rfc791.txt |
| SUSv2 | CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606) | http://www.opengroup.org/publications/catalog/un.htm |
| SUSv2 Commands and Utilities | The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604) | http://www.opengroup.org/publications/catalog/un.htm |
| SVID Issue 3 | American Telephone and Telegraph Company, System V Interface Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989.(ISBN 0201566524) | |
| SVID Issue 4 | System V Interface Definition, Fourth Edition | |
| System V ABI | System V Application Binary Interface, Edition 4.1 | http://www.caldera.com/developers/devspecs/gabi41.pdf |
| System V ABI Update | System V Application Binary Interface - DRAFT - 17 December 2003 | http://www.caldera.com/developers/gabi/2003-12-17/contents.html |
| this specification | Linux Standard Base | http://www.linuxbase.org/spec/ |
| X/Open 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

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

15

16

17

18

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 | LI18NUNIX 2000 Globalization Specification, Version 1.0 with Amendment 4 | http://www.li18nux.org/docs/html/LI18NUNIX-2000-amd4.htm |
| Linux Allocated Device Registry | LINUX ALLOCATED DEVICES | http://www.lanana.org/docs/device-list/devices.txt |
| PAM | Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar & R.Schemers (SunSoft) | http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt |
| RFC 1321: The MD5 Message-Digest Algorithm | IETF RFC 1321: The MD5 Message-Digest Algorithm | http://www.ietf.org/rfc/rfc1321.txt |
| RFC 1831/1832 RPC & XDR | IETF RFC 1831 & 1832 | http://www.ietf.org/ |
| RFC 1833: Binding Protocols for ONC RPC Version 2 | IETF RFC 1833: Binding Protocols for ONC RPC Version 2 | http://www.ietf.org/rfc/rfc1833.txt |
| RFC 1950: ZLIB Compressed Data Format Specification | IETF RFC 1950: ZLIB Compressed Data Format Specification | http://www.ietf.org/rfc/rfc1950.txt |

| Name | Title | URL |
|--|---|---|
| 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 IA64 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.6.1 |
| libdl | libdl.so.2 |
| libcrypt | libcrypt.so.1 |
| libz | libz.so.1 |
| libncurses | libncurses.so.5 |
| libutil | libutil.so.1 |
| libc | libc.so.6.1 |
| libpthread | libpthread.so.0 |
| proginterp | /lib/ld-lsb-ia64.so.3 |
| libgcc_s | libgcc_s.so.1 |

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

3.2 LSB Implementation Conformance

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

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

15 A conforming implementation shall satisfy the following requirements:

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

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

3.3 LSB Application Conformance

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

56 A conforming application shall satisfy the following requirements:

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

3 Requirements

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

4 Definitions

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

5 Terminology

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

2 archLSB

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

6 Binary Standard

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

9 gLSB

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

12 implementation-defined

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

20 Shell Script

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

23 Source Standard

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

26 undefined

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

33 unspecified

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

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

6 Documentation Conventions

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

2 `function()`

3 the name of a function

4 **command**

5 the name of a command or utility

6 `CONSTANT`

7 a constant value

8 *parameter*

9 a parameter

10 `variable`

11 a variable

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

14 `name`

15 the name of the interface

16 `(symver)`

17 An optional symbol version identifier, if required.

18 `[refno]`

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

21 For example,

22 `forkpty(GLIBC_2.0) [1SUSv3]`

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

25 **Note:** Symbol versions are defined in the ~~listed references below the~~
26 ~~table~~-architecture specific supplements only.

II Executable and Linking Format (ELF)

7 Introduction

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

8 Low Level System Information

8.1 Machine Interface

8.1.1 Processor Architecture

The Itanium™ Architecture is specified by the following documents

- Itanium™ Architecture Software Developer's Manual Volume 1
- Itanium™ Architecture Software Developer's Manual Volume 2
- Itanium™ Architecture Software Developer's Manual Volume 3
- Itanium™ Architecture Software Developer's Manual Volume 4
- Itanium™ Software Conventions and Runtime Guide
- Intel® Itanium™ Processor-specific Application Binary Interface

Only the features of the Itanium™ processor instruction set may be assumed to be present. An application should determine if any additional instruction set features are available before using those additional features. If a feature is not present, then the application may not use it.

~~Only Conforming applications may use only~~ instructions which do not require elevated privileges ~~may be used by~~.

~~Conforming applications shall not invoke the application.~~

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

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

There are some features of the Itanium™ processor architecture that need not be supported by a conforming implementation. These are described in this chapter. A conforming application shall not rely on these features.

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

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

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

This specification describes only LP64 (i.e. 32-bit integers, 64-bit longs and pointers) based implementations. Implementations may also provide ILP32 (32-bit integers, longs, and pointers), but conforming applications shall not rely on support for ILP32. See section 1.2 of the Intel® Itanium™ Processor-specific Application Binary Interface for further information.

8.1.2 Data Representation

36 The following sections, in conjunction with section 4 of Itanium™ Software
 37 Conventions and Runtime Guide, define the size, alignment requirements, and
 38 hardware representation of the standard C data types.

39 Within this specification, the term `byte` refers to an 8-bit object, the term `halfword`
 40 refers to a 16-bit object, the term `word` refers to a 32-bit object, the term `doubleword`
 41 refers to a 64-bit object, and the term `quadword` refers to a 128-bit object.

8.1.2.1 Byte Ordering

42
 43 LSB-conforming applications shall use little-endian byte ordering. LSB-conforming
 44 implementations may support big-endian applications.

8.1.2.2 Fundamental Types

45
 46 Table 8-1 describes how fundamental C language data types shall be represented:

47 **Table 8-1 Scalar Types**

| Type | C | sizeof | Alignment (bytes) | Hardware Representation |
|----------|------------------|--------|-------------------|-------------------------|
| Integral | _Bool | 1 | 1 | byte (sign unspecified) |
| | char | 1 | 1 | signed byte |
| | signed char | | | |
| | unsigned char | | | signed byte |
| | short | 2 | 2 | signed halfword |
| | signed short | | | |
| | unsigned short | | | unsigned halfword |
| | int | 4 | 4 | signed word |
| | signed int | | | |
| | unsigned int | | | unsigned word |
| | long | 8 | 8 | signed doubleword |
| | signed long | | | |
| | unsigned long | | | unsigned doubleword |
| | long long | 8 | 8 | signed doubleword |
| | signed long long | | | |

| Type | C | sizeof | Alignment (bytes) | Hardware Representation |
|----------------|------------------------|--------|-------------------|-------------------------|
| | unsigned long long | | | unsigned doubleword |
| Pointer | <i>any-type</i> * | 8 | 8 | unsigned doubleword |
| | <i>any-type</i> (*)() | | | |
| Floating-Point | float | 4 | 4 | IEEE Single-precision |
| | double | 8 | 8 | IEEE Double-precision |
| | long double | 16 | 16 | IEEE Double-extended |

48

49

A null pointer (for all types) shall have the value zero.

50

8.1.2.3 Aggregates and Unions

51

52

53

54

55

Aggregates (structures and arrays) and unions assume the alignment of their most strictly aligned component. The size of any object, including aggregates and unions, shall always be a multiple of the object's alignment. An array uses the same alignment as its elements. Structure and union objects may require padding to meet size and element constraints. The contents of such padding is undefined.

56

57

- An entire structure or union object shall be aligned on the same boundary as its most strictly aligned member.
- Each member shall be assigned to the lowest available offset with the appropriate alignment. This may require *internal padding*, depending on the previous member.
- A structure's size shall be increased, if necessary, to make it a multiple of the alignment. This may require *tail padding*, depending on the last member.

58

59

60

61

62

A conforming application shall not read padding.

63

| <pre> struct { char c; } </pre> | |
|-------------------------------------|--------|
| Byte aligned, sizeof is 1 | |
| Offset | Byte 0 |
| 0 | c^0 |

64

65

Figure 8-1 Structure Smaller Than A Word

| |
|---|
| <pre> struct { char c; char d; short s; int i; } </pre> |
|---|

| | | | | |
|----------------------------------|----------------|---------------|----------------|----------------|
| <pre> long l; } </pre> | | | | |
| Doubleword Aligned, sizeof is 16 | | | | |
| Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
| 0 | s ² | | d ¹ | c ⁰ |
| 4 | i ⁰ | | | |
| 8 | l ⁰ | | | |
| 12 | | | | |

Figure 8-2 No Padding

| | | | | |
|---|------------------|---------------|----------------|----------------|
| <pre> struct { char c; long l; int i; short s; } </pre> | | | | |
| Doubleword Aligned, sizeof is 24 | | | | |
| Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
| 0 | pad ¹ | | | c ⁰ |
| 4 | pad ¹ | | | |
| 8 | l ⁰ | | | |
| 12 | | | | |
| 16 | i ⁰ | | | |
| 20 | pad ² | | s ⁰ | |

Figure 8-3 Internal and Tail Padding

8.1.2.4 Bit Fields

C `struct` and `union` definitions may have *bit-fields*, which define integral objects with a specified number of bits.

Bit fields that are declared with neither `signed` nor `unsigned` specifier shall always be treated as `unsigned`. Bit fields obey the same size and alignment rules as other structure and union members, with the following additional properties:

- Bit-fields are allocated from right to left (least to most significant).
- A bit-field must entirely reside in a storage unit for its appropriate type. A bit field shall never cross its unit boundary.
- Bit-fields may share a storage unit with other `struct/union` members, including members that are not bit fields. Such other `struct/union` members shall occupy different parts of the storage unit.

- 84 • The type of unnamed bit-fields shall not affect the alignment of a structure or
 85 union, although individual bit-field member offsets shall obey the alignment
 86 constraints.

| Bit-field Type | Width w | Range |
|---|-----------|---|
| signed char char unsigned char | 1 to 8 | -2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1 |
| signed short short unsigned short | 1 to 16 | -2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1 |
| signed int int unsigned int | 1 to 32 | -2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1 |
| signed long long unsigned long | 1 to 64 | -2^{w-1} to $2^{w-1}-1$ 0 to 2^w-1 0 to 2^w-1 |

87

88

Figure 8-4 Bit-Field Ranges

8.2 Function Calling Sequence

89 LSB-conforming applications shall use the procedure linkage and function calling
 90 sequence as defined in Chapter 8.4 of the Itanium™ Software Conventions and
 91 Runtime Guide.

8.2.1 Registers

92 The CPU general and other registers are as defined in the Itanium™ Architecture
 93 Software Developer's Manual Volume 1 Section 3.1.

8.2.2 Floating Point Registers

94 The floating point registers are as defined in the Itanium™ Architecture Software
 95 Developer's Manual Volume 1 Section 3.1.

8.2.3 Stack Frame

96 The stackframe layout is as described in the Itanium™ Software Conventions and
 97 Runtime Guide Chapter 8.4.

8.2.4 Arguments

8.2.4.1 Introduction

98 The procedure parameter passing mechanism is as described in the Itanium™
 99 Software Conventions and Runtime Guide Chapter 8.5. The following subsections
 100 provide additional information.
 101

8.2.4.2 Integral/Pointer

102 See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.
 103

104 **8.2.4.3 Floating Point**

105 See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

106 **8.2.4.4 Struct and Union Point**

107 See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

108 **8.2.4.5 Variable Arguments**

109 See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.4.

8.2.5 Return Values

110 **8.2.5.1 Introduction**

111 Values are returned from functions as described in Itanium™ Software Conventions
112 and Runtime Guide Chapter 8.6, and as further described here.

113 **8.2.5.2 Void**

114 Functions that return no value (void functions) are not required to put any
115 particular value in any general register.

116 **8.2.5.3 Integral/Pointer**

117 See Itanium™ Software Conventions and Runtime Guide Chapter 8.6.

118 **8.2.5.4 Floating Point**

119 See Itanium™ Software Conventions and Runtime Guide Chapter 8.6.

120 **8.2.5.5 Struct and Union**

121 See Itanium™ Software Conventions and Runtime Guide Chapter 8.6 (aggregate
122 return values). Depending on the size (including any padding), aggregate data types
123 may be passed in one or more general registers, or in memory.

8.3 Operating System Interface

124 LSB-conforming applications shall use the Operating System Interfaces as defined in
125 Chapter 3 of the Intel® Itanium™ Processor-specific Application Binary Interface.

8.3.1 Processor Execution Mode

126 Applications must assume that they will execute in the least privileged user mode
127 (i.e. level 3). Other privilege levels are reserved for the Operating System.

8.3.2 Exception Interface

128 **8.3.2.1 Introduction**

129 LSB-conforming implementations shall support the exception interface as specified
130 in Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.1.

131 **8.3.2.2 Hardware Exception Types**

132 See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.1.

133 **8.3.2.3 Software Trap Types**

134 See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.1.

8.3.3 Signal Delivery

135 LSB-conforming systems shall deliver signals as specified in Intel® Itanium™
136 Processor-specific Application Binary Interface, section 3.3.2.

8.3.3.1 Signal Handler Interface

137
138 The signal handler interface shall be as specified in Intel® Itanium™
139 Processor-specific Application Binary Interface, section 3.3.3.

8.3.4 Debugging Support

140 The LSB does not specify debugging information.

8.3.5 Process Startup

141 LSB-conforming systems shall initialize processes as specified in Intel® Itanium™
142 Processor-specific Application Binary Interface, section 3.3.5.

8.4 Process Initialization

143 LSB-conforming applications shall use the Process Startup as defined in Section 3.3.5
144 of the Intel® Itanium™ Processor-specific Application Binary Interface.

8.4.1 Special Registers

145 Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.5,
146 defines required register initializations for process startup.

8.4.2 Process Stack (on entry)

147 As defined in Intel® Itanium™ Processor-specific Application Binary Interface,
148 section 3.3.5, the return pointer register (rp) shall contain a valid return address,
149 such that if the application program returns from the main entry routine, the
150 implementation shall cause the application to exit normally, using the returned
151 value as the exit status. Further, the unwind information for this "bottom of stack"
152 routine in the implementation shall provide a mechanism for recognizing the bottom
153 of the stack during a stack unwind.

8.4.3 Auxiliary Vector

154 The auxiliary vector conveys information from the operating system to the
155 application. Only the terminating null auxiliary vector entry is required, but if any
156 other entries are present, they shall be interpreted as follows. This vector is an array
157 of the following structures.

```
158 typedef struct
159 {
160     long int a_type;           /* Entry type */
161     union
162     {
163         long int a_val;       /* Integer value */
164         void *a_ptr;          /* Pointer value */
165         void (*a_fcn) (void); /* Function pointer value */
166     } a_un;
167 } auxv_t;
```

168 The application shall interpret the a_un value according to the a_type. Other
169 auxiliary vector types are reserved.

170 The `a_type` field shall contain one of the following values:
171 AT_NULL
172 The last entry in the array has type AT_NULL. The value in `a_un` is undefined.
173 AT_IGNORE
174 The value in `a_un` is undefined, and should be ignored.
175 AT_EXECD
176 File descriptor of program
177 AT_PHDR
178 Program headers for program
179 AT_PHENT
180 Size of program header entry
181 AT_PHNUM
182 Number of program headers
183 AT_PAGESZ
184 System page size
185 AT_BASE
186 Base address of interpreter
187 AT_FLAGS
188 Flags
189 AT_ENTRY
190 Entry point of program
191 AT_NOTELF
192 Program is not ELF
193 AT_UID
194 Real uid
195 AT_EUID
196 Effective uid
197 AT_GID
198 Real gid
199 AT_EGID
200 Effective gid
201 AT_CLKTCK
202 Frequency of `times()`

| | |
|-----|--|
| 203 | AT_PLATFORM |
| 204 | String identifying platform. |
| 205 | AT_HWCAP |
| 206 | Machine dependent hints about processor capabilities. |
| 207 | AT_FPUCW |
| 208 | Used FPU control word |
| 209 | AT_DCACHEBSIZE |
| 210 | Data cache block size |
| 211 | AT_ICACHEBSIZE |
| 212 | Instruction cache block size |
| 213 | AT_UCACHEBSIZE |
| 214 | Unified cache block size |
| 215 | Note: The auxiliary vector is intended for passing information from the operating |
| 216 | system to the program interpreter. |

8.4.4 Environment

217 Although a pointer to the environment vector should be available as a third
 218 argument to the `main()` entry point, conforming applications should use `getenv()`
 219 to access the environment. (See ISO POSIX (2003), Section `exec()`).

8.5 Coding Examples

8.5.1 Introduction

220 LSB-conforming applications may implement fundamental operations using the
 221 Coding Examples as shown below.
 222 Sample code sequences and coding conventions can be found in Itanium™ Software
 223 Conventions and Runtime Guide, Chapter 9.

8.5.2 Code Model Overview/Architecture Constraints

224 As defined in Intel® Itanium™ Processor-specific Application Binary Interface,
 225 relocatable files, executable files, and shared object files that are supplied as part of
 226 an application shall use Position Independent Code, as described in Itanium™
 227 Software Conventions and Runtime Guide, Chapter 12.

8.5.3 Position-Independent Function Prologue

228 See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.

8.5.4 Data Objects

229 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4,
 230 and Itanium™ Software Conventions and Runtime Guide, Chapter 12.3.

8.5.4.1 Absolute Load & Store

231 Conforming applications shall not use absolute addressing.
 232

233 **8.5.4.2 Position Relative Load & Store**

234 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4.

8.5.5 Function Calls

235 See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.

236 Four types of procedure call are defined in Itanium™ Software Conventions and
237 Runtime Guide, Chapter 8.3. Although special calling conventions are permitted,
238 provided that the compiler and runtime library agree on these conventions, none are
239 defined for this standard. Consequently, no application shall depend on a type of
240 procedure call other than Direct Calls, Direct Dynamically Linked Calls, or Indirect
241 Calls, as defined in Itanium™ Software Conventions and Runtime Guide, Chapter
242 8.3.

243 **8.5.5.1 Absolute Direct Function Call**

244 Conforming applications shall not use absolute addressing.

245 **8.5.5.2 Absolute Indirect Function Call**

246 Conforming applications shall not use absolute addressing.

247 **8.5.5.3 Position-Independent Direct Function Call**

248 See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.1.

249 **8.5.5.4 Position-Independent Indirect Function Call**

250 See Itanium™ Software Conventions and Runtime Guide, Chapter 8.4.2.

8.5.6 Branching

251 Branching is described in Itanium™ Architecture Software Developer's Manual
252 Volume 4, Chapter 4.5.

253 **8.5.6.1 Branch Instruction**

254 See Itanium™ Architecture Software Developer's Manual Volume 4, Chapter 4.5.

255 **8.5.6.2 Absolute switch() code**

256 Conforming applications shall not use absolute addressing.

257 **8.5.6.3 Position-Independent switch() code**

258 Where there are several possible targets for a branch, the compiler may use a
259 number of different code generation strategies. See Itanium™ Software
260 Conventions and Runtime Guide, Chapter 9.1.7.

8.6 C Stack Frame

8.6.1 Variable Argument List

261 See Itanium™ Software Conventions and Runtime Guide, Chapter 8.5.2, and 8.5.4.

8.6.2 Dynamic Allocation of Stack Space

262 The C library `alloca()` function should be used to dynamically allocate stack space.

8.7 Debug Information

263

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, Intel® Itanium™
3 Processor-specific Application Binary Interface and as supplemented by the Linux
4 Standard Base 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 Intel®
6 Itanium™ Processor-specific Application Binary Interface, Chapter 4.
7 Implementations shall support the LP64 model. It is unspecified whether or not the
8 ILP32 model shall also be supported.

9.2.1.1 File Class

9
10 For LP64 relocatable objects, the file class value in `e_ident[EI_CLASS]` may be
11 either `ELFCLASS32` or `ELFCLASS64`, and a conforming linker must be able to
12 process either or both classes.

9.2.1.2 Data Encoding

13
14 Implementations shall support 2's complement, little endian data encoding. The data
15 encoding value in `e_ident[EI_DATA]` shall contain the value `ELFDATA2LSB`.

9.2.1.3 OS Identification

16
17 The OS Identification field `e_ident[EI_OSABI]` shall contain the value
18 `ELFOSABI_NONE`.

9.2.1.4 Processor Identification

19
20 The processor identification value held in `e_machine` shall contain the value
21 `EM_IA_64`.

9.2.1.5 Processor Specific Flags

22
23 The flags field `e_flags` shall be as described in Intel® Itanium™ Processor-specific
24 Application Binary Interface, Chapter 4.1.1.6.

25 The following additional processor-specific flags are defined:

26 **Table 9-1 Additional Processor-Specific Flags**

| Name | Value |
|--|-------------------------|
| <code>EF_IA_64_LINUX_EXECUTABLE_STACK</code> <code>ACK</code> | <code>0x00000001</code> |

27
28 `EF_IA_64_LINUX_EXECUTABLE_STACK`

29 The stack and heap sections are executable. If this flag is not set, code can not be
30 executed from the stack or heap.

9.3 Sections

31 The Itanium™ architecture defines two processor-specific section types, as described
 32 in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.

9.3.1 Special Sections

33 The following sections are defined in the Intel® Itanium™ Processor-specific
 34 Application Binary Interface.

35 **Table 9-2 ELF Special Sections**

| Name | Type | Attributes |
|--------------------|------------------|-------------------------------------|
| .got | SHT_PROGBITS | SHF_ALLOC+SHF_WRITE+SHF_IA_64_SHORT |
| .IA_64.archext | SHT_IA_64_EXT | 0 |
| .IA_64.pltoff | SHT_PROGBITS | SHF_ALLOC+SHF_WRITE+SHF_IA_64_SHORT |
| .IA_64.unwind | SHT_IA_64_UNWIND | SHF_ALLOC+SHF_LINK_ORDER |
| .IA_64.unwind_info | SHT_PROGBITS | SHF_ALLOC |
| .plt | SHT_PROGBITS | SHF_ALLOC+SHF_EXECINSTR |
| .sbss | SHT_NOBITS | SHF_ALLOC+SHF_WRITE+SHF_IA_64_SHORT |
| .sdata | SHT_PROGBITS | SHF_ALLOC+SHF_WRITE+SHF_IA_64_SHORT |
| .sdata1 | SHT_PROGBITS | SHF_ALLOC+SHF_WRITE+SHF_IA_64_SHORT |

36

37 .got

38 This section holds the Global Offset Table. See 'Coding Examples' in Chapter 3,
 39 'Special Sections' in Chapter 4, and 'Global Offset Table' in Chapter 5 of the
 40 processor supplement for more information.

41 .IA_64.archext

42 This section holds product-specific extension bits. The link editor will perform a
 43 logical "or" of the extension bits of each object when creating an executable so
 44 that it creates only a single .IA_64.archext section in the executable.

45 .IA_64.pltoff

46 This section holds local function descriptor entries.

47 .IA_64.unwind

48 This section holds the unwind function table. The contents are described in the
 49 Intel (r) Itanium (tm) Processor Specific ABI.

50 `.IA_64.unwind_info`
 51 This section holds stack unwind and and exception handling information. The
 52 exception handling information is programming language specific, and is
 53 unspecified.

54 `.plt`
 55 This section holds the Procedure Linkage Table.

56 `.sbss`
 57 This section holds uninitialized data that contribute to the program's memory
 58 image. Data objects contained in this section are recommended to be eight bytes
 59 or less in size. The system initializes the data with zeroes when the program
 60 begins to run. The section occupies no file space, as indicated by the section type
 61 SHT_NOBITS. The `.sbss` section is placed so it may be accessed using short
 62 direct addressing (22 bit offset from `gp`).

63 `.sdata`
 64 This section and the `.sdata1` section hold initialized data that contribute to the
 65 program's memory image. Data objects contained in this section are
 66 recommended to be eight bytes or less in size. The `.sdata` and `.sdata1` sections
 67 are placed so they may be accessed using short direct addressing (22 bit offset
 68 from `gp`).

69 `.sdata1`
 70 See `.sdata`.

9.3.2 Linux Special Sections

71 The following Linux IA-64 specific sections are defined here.

72 **Table 9-3 Additional Special Sections**

| Name | Type | Attributes |
|---------------------------------|--------------|------------|
| <code>.opd</code> | SHT_PROGBITS | SHF_ALLOC |
| <code>.rela.dyn</code> | SHT_RELA | SHF_ALLOC |
| <code>.rela.IA_64.pltoff</code> | SHT_RELA | SHF_ALLOC |

73

74 `.opd`
 75 This section holds function descriptors

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

79 `.rela.IA_64.pltoff`
 80 This section holds relocation information, as described in 'Relocation'. These
 81 relocations are applied to the `.IA_64.pltoff` section.

9.3.3 Section Types

82 Section Types are described in the Intel® Itanium™ Processor-specific Application
 83 Binary Interface, Chapter 4.2. LSB conforming implementations are not required to
 84 use any sections in the range from `SHT_IA_64_LOPSREG` to `SHT_IA_64_HIPSREG`.
 85 Additionally, LSB conforming implementations are not required to support the
 86 `SHT_IA_64_PRIORITY_INIT` section, beyond the gABI requirements for the handling
 87 of unrecognized section types, linking them into a contiguous section in the object
 88 file created by the static linker.

9.3.4 Section Attribute Flags

89 LSB-conforming implementations shall support the section attribute flags specified
 90 in Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.2.2.

9.3.5 Special Section Types

91 The special section types `SHT_IA64_EXT` and `SHT_IA64_UNWIND` are defined in Intel®
 92 Itanium™ Processor-specific Application Binary Interface, Chapter 4.2.1.

9.4 Symbol Table

93 If an executable file contains a reference to a function defined in one of its associated
 94 shared objects, the symbol table section for that file shall contain an entry for that
 95 symbol. The `st_shndx` member of that symbol table entry contains `SHN_UNDEF`. This
 96 signals to the dynamic linker that the symbol definition for that function is not
 97 contained in the executable file itself. If that symbol has been allocated a procedure
 98 linkage table entry in the executable file, and the `st_value` member for that symbol
 99 table entry is non-zero, the value shall contain the virtual address of the first
 100 instruction of that procedure linkage table entry. Otherwise, the `st_value` member
 101 contains zero. This procedure linkage table entry address is used by the dynamic
 102 linker in resolving references to the address of the function.

9.5 Relocation

9.5.1 Relocation Types

103 LSB-conforming systems shall support the relocation types described in Intel®
 104 Itanium™ Processor-specific Application Binary Interface, Chapter 4.3.

10 Program Loading and Dynamic Linking

10.1 Introduction

1 LSB-conforming implementations shall support the object file information and
2 system actions that create running programs as specified in the System V ABI, Intel®
3 Itanium™ Processor-specific Application Binary Interface and as supplemented by
4 the Linux Standard Base Specification and this document.

10.2 Program Header

5 The program header shall be as defined in the Intel® Itanium™ Processor-specific
6 Application Binary Interface, Chapter 5.

10.2.1 Types

7 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.1.

10.2.2 Flags

8 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.1.

10.3 Program Loading

9 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.2.

10.4 Dynamic Linking

10 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.

10.4.1 Dynamic Entries

10.4.1.1 ELF Dynamic Entries

11 The following dynamic entries are defined in the Intel® Itanium™
12 Processor-specific Application Binary Interface, Chapter 5.3.2.

13 DT_PLTGOT

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

10.4.1.2 Additional Dynamic Entries

17 The following dynamic entries are defined here.

18 DT_RELACOUNT

19 The number of relative relocations in `.rela.dyn`
20

10.4.2 Global Offset Table

21 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4.

10.4.3 Shared Object Dependencies

22 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.3.

10.4.4 Function Addresses

23 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.5.

10.4.5 Procedure Linkage Table

24 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.6.

10.4.6 Initialization and Termination Functions

25 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.7.

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 Itanium™ platform are defined here.
 5 This section should be used in conjunction with the corresponding section in the
 6 Linux Standard Base Specification.

11.1 Program Interpreter/Dynamic Linker

7 The ~~LSB specifies the~~ Program Interpreter ~~shall~~ be /lib/ld-lsb-ia64.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.6.1 |

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

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

11.2.1 RPC

11.2.1.1 Interfaces for RPC

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

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

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

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

19
20
21
22
23

Referenced Specification(s)

[1]. SVID Issue 4

[2]. SVID Issue 3

[3]. this specification

| | | | |
|--|--|---|--|
| <code>authnone_create(GLIBC_2.2)</code> [SVID.4] | <code>clnt_create(GLIBC_2.2)</code> [SVID.4] | <code>clnt_pcreateerror(GLIBC_2.2)</code> [SVID.4] | <code>clnt_permno(GLIBC_2.2)</code> [SVID.4] |
| <code>clnt_perror(GLIBC_2.2)</code> [SVID.4] | <code>clnt_sprecreateerror(GLIBC_2.2)</code> [SVID.4] | <code>clnt_sperrno(GLIBC_2.2)</code> [SVID.4] | <code>clnt_sperror(GLIBC_2.2)</code> [SVID.4] |
| <code>key_decryptsession(GLIBC_2.2)</code> [SVID.3] | <code>pmap_getport(GLIBC_2.2)</code> [LSB] | <code>pmap_set(GLIBC_2.2)</code> [LSB] | <code>pmap_unset(GLIBC_2.2)</code> [LSB] |
| <code>svc_getreqset(GLIBC_2.2)</code> [SVID.3] | <code>svc_register(GLIBC_2.2)</code> [LSB] | <code>svc_run(GLIBC_2.2)</code> [LSB] | <code>svc_sendreply(GLIBC_2.2)</code> [LSB] |
| <code>svcerr_auth(GLIBC_2.2)</code> [SVID.3] | <code>svcerr_decode(GLIBC_2.2)</code> [SVID.3] | <code>svcerr_noproc(GLIBC_2.2)</code> [SVID.3] | <code>svcerr_noprog(GLIBC_2.2)</code> [SVID.3] |
| <code>svcerr_progvers(GLIBC_2.2)</code> | <code>svcerr_systemerr(GLIBC_2.2)</code> | <code>svcerr_weakauth(GLIBC_2.2)</code> | <code>svctcp_create(GLIBC_2.2)</code> |

| | | | |
|---------------------------------|--|--|-------------------------------------|
| GLIBC_2.2) [SVID.3] | GLIBC_2.2) [SVID.3] | GLIBC_2.2) [SVID.3] | BC_2.2) [LSB] |
| svcdp_create(GLIBC_2.2) [LSB] | xdr_accepted_repl y(GLIBC_2.2) [SVID.3] | xdr_array(GLIBC_2.2) [SVID.3] | xdr_bool(GLIBC_2.2) [SVID.3] |
| xdr_bytes(GLIBC_2.2) [SVID.3] | xdr_callhdr(GLIBC_2.2) [SVID.3] | xdr_callmsg(GLIBC_2.2) [SVID.3] | xdr_char(GLIBC_2.2) [SVID.3] |
| xdr_double(GLIBC_2.2) [SVID.3] | xdr_enum(GLIBC_2.2) [SVID.3] | xdr_float(GLIBC_2.2) [SVID.3] | xdr_free(GLIBC_2.2) [SVID.3] |
| xdr_int(GLIBC_2.2) [SVID.3] | xdr_long(GLIBC_2.2) [SVID.3] | xdr_opaque(GLIBC_2.2) [SVID.3] | xdr_opaque_auth(GLIBC_2.2) [SVID.3] |
| xdr_pointer(GLIBC_2.2) [SVID.3] | xdr_reference(GLIBC_2.2) [SVID.3] | xdr_rejected_repl y(GLIBC_2.2) [SVID.3] | xdr_replymsg(GLIBC_2.2) [SVID.3] |
| xdr_short(GLIBC_2.2) [SVID.3] | xdr_string(GLIBC_2.2) [SVID.3] | xdr_u_char(GLIBC_2.2) [SVID.3] | xdr_u_int(GLIBC_2.2) [LSB] |
| xdr_u_long(GLIBC_2.2) [SVID.3] | xdr_u_short(GLIBC_2.2) [SVID.3] | xdr_union(GLIBC_2.2) [SVID.3] | xdr_vector(GLIBC_2.2) [SVID.3] |
| xdr_void(GLIBC_2.2) [SVID.3] | xdr_wrapstring(GLIBC_2.2) [SVID.3] | xdrmem_create(GLIBC_2.2) [SVID.3] | xdrrec_create(GLIBC_2.2) [SVID.3] |
| xdrrec_eof(GLIBC_2.2) [SVID.3] | | | |

24

11.2.2 System Calls

25

11.2.2.1 Interfaces for System Calls

26

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

27

28

29

Table 11-3 libc - System Calls Function Interfaces

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

| | | | | |
|--------------------------------------|---|--|--|--------------------------------------|
| <code>_-2.2)-[2]</code> | <code>_-2)-[2]</code> | <code>€_-2.2)-[2]</code> | <code>€_-2.2)-[2]</code> | <code>_-2.2)-[2]</code> |
| <code>aect(GLIBC_2.2)-[1]</code> | <code>fstatvfs(GLIBC_2.2)-[2]</code> | <code>link(GLIBC_2.2)-[1]</code> | <code>rmdir(GLIBC_2.2)-[2]</code> | <code>stime(GLIBC_2.2)-[1]</code> |
| <code>alarm(GLIBC_2.2)-[2]</code> | <code>fsync(GLIBC_2.2)-[2]</code> | <code>lockf(GLIBC_2.2)-[2]</code> | <code>sbrk(GLIBC_2.2)-[4]</code> | <code>symlink(GLIBC_2.2)-[2]</code> |
| <code>brk(GLIBC_2.2)-[4]</code> | <code>ftime(GLIBC_2.2)-[2]</code> | <code>lseek(GLIBC_2.2)-[2]</code> | <code>sched_get_priority_max(GLIBC_2.2)-[2]</code> | <code>sync(GLIBC_2.2)-[2]</code> |
| <code>chdir(GLIBC_2.2)-[2]</code> | <code>ftruncate(GLIBC_2.2)-[2]</code> | <code>mkdir(GLIBC_2.2)-[2]</code> | <code>sched_get_priority_min(GLIBC_2.2)-[2]</code> | <code>sysconf(GLIBC_2.2)-[2]</code> |
| <code>chmod(GLIBC_2.2)-[2]</code> | <code>getcontext(GLIBC_2.2)-[2]</code> | <code>mknfif(GLIBC_2.2)-[2]</code> | <code>sched_getparam(GLIBC_2.2)-[2]</code> | <code>time(GLIBC_2.2)-[2]</code> |
| <code>chown(GLIBC_2.2)-[2]</code> | <code>getegid(GLIBC_2.2)-[2]</code> | <code>mlock(GLIBC_2.2)-[2]</code> | <code>sched_getscheduler(GLIBC_2.2)-[2]</code> | <code>times(GLIBC_2.2)-[2]</code> |
| <code>chroot(GLIBC_2.2)-[4]</code> | <code>geteuid(GLIBC_2.2)-[2]</code> | <code>mlockall(GLIBC_2.2)-[2]</code> | <code>sched_rr_get_interval(GLIBC_2.2)-[2]</code> | <code>truncate(GLIBC_2.2)-[2]</code> |
| <code>clock(GLIBC_2.2)-[2]</code> | <code>getgid(GLIBC_2.2)-[2]</code> | <code>mmap(GLIBC_2.2)-[2]</code> | <code>sched_setparam(GLIBC_2.2)-[2]</code> | <code>ulimit(GLIBC_2.2)-[2]</code> |
| <code>close(GLIBC_2.2)-[2]</code> | <code>getgroups(GLIBC_2.2)-[2]</code> | <code>mprotect(GLIBC_2.2)-[2]</code> | <code>sched_setscheduler(GLIBC_2.2)-[2]</code> | <code>umask(GLIBC_2.2)-[2]</code> |
| <code>closedir(GLIBC_2.2)-[2]</code> | <code>getitimer(GLIBC_2.2)-[2]</code> | <code>msync(GLIBC_2.2)-[2]</code> | <code>sched_yield(GLIBC_2.2)-[2]</code> | <code>uname(GLIBC_2.2)-[2]</code> |
| <code>creat(GLIBC_2.2)-[2]</code> | <code>getloadavg(GLIBC_2.2)-[1]</code> | <code>munlock(GLIBC_2.2)-[2]</code> | <code>select(GLIBC_2.2)-[2]</code> | <code>unlink(GLIBC_2.2)-[1]</code> |
| <code>dup(GLIBC_2.2)-[2]</code> | <code>getpagesize(GLIBC_2.2)-[4]</code> | <code>munlockall(GLIBC_2.2)-[2]</code> | <code>setcontext(GLIBC_2.2)-[2]</code> | <code>utime(GLIBC_2.2)-[2]</code> |
| <code>dup2(GLIBC_2.2)-[2]</code> | <code>getpgid(GLIBC_2.2)-[2]</code> | <code>munmap(GLIBC_2.2)-[2]</code> | <code>setegid(GLIBC_2.2)-[2]</code> | <code>utimes(GLIBC_2.2)-[2]</code> |
| <code>execl(GLIBC_2.2)-[2]</code> | <code>getpgrp(GLIBC_2.2)-[2]</code> | <code>nanosleep(GLIBC_2.2)-[2]</code> | <code>seteuid(GLIBC_2.2)-[2]</code> | <code>vfork(GLIBC_2.2)-[2]</code> |
| <code>execl(GLIBC_2.2)-[2]</code> | <code>getpid(GLIBC_2.2)-[2]</code> | <code>nice(GLIBC_2.2)-[2]</code> | <code>setgid(GLIBC_2.2)-[2]</code> | <code>wait(GLIBC_2.2)-[2]</code> |
| <code>execlp(GLIBC_2.2)-[2]</code> | <code>getppid(GLIBC_2.2)-[2]</code> | <code>open(GLIBC_2.2)-[2]</code> | <code>setitimer(GLIBC_2.2)-[2]</code> | <code>wait4(GLIBC_2.2)-[1]</code> |
| <code>execv(GLIBC_2.2)-[2]</code> | <code>getpriority(GLIBC_2.2)-[2]</code> | <code>opendir(GLIBC_2.2)-[2]</code> | <code>setpgid(GLIBC_2.2)-[2]</code> | <code>waitpid(GLIBC_2.2)-[1]</code> |

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

30

31

Referenced Specification(s)

32

[1]. this specification

33

[2]. ISO POSIX (2003)

34

[3]. Large File Support

35

[4]. SUSv2

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

| | | | |
|---------------------------------------|---|---|---------------------------------------|
| [SUSv3] |) [SUSv3] |) [SUSv3] | _2.2) [LSB] |
| ioctl(GLIBC_2.2) [LSB] | kill(GLIBC_2.2) [LSB] | killpg(GLIBC_2.2) [SUSv3] | lchown(GLIBC_2.2) [SUSv3] |
| link(GLIBC_2.2) [LSB] | lockf(GLIBC_2.2) [SUSv3] | lseek(GLIBC_2.2) [SUSv3] | mkdir(GLIBC_2.2) [SUSv3] |
| mkfifo(GLIBC_2.2) [SUSv3] | mlock(GLIBC_2.2) [SUSv3] | mlockall(GLIBC_2.2) [SUSv3] | mmap(GLIBC_2.2) [SUSv3] |
| mprotect(GLIBC_2.2) [SUSv3] | msync(GLIBC_2.2) [SUSv3] | munlock(GLIBC_2.2) [SUSv3] | munlockall(GLIBC_2.2) [SUSv3] |
| munmap(GLIBC_2.2) [SUSv3] | nanosleep(GLIBC_2.2) [SUSv3] | nice(GLIBC_2.2) [SUSv3] | open(GLIBC_2.2) [SUSv3] |
| opendir(GLIBC_2.2) [SUSv3] | pathconf(GLIBC_2.2) [SUSv3] | pause(GLIBC_2.2) [SUSv3] | pipe(GLIBC_2.2) [SUSv3] |
| poll(GLIBC_2.2) [SUSv3] | read(GLIBC_2.2) [SUSv3] | readdir(GLIBC_2.2) [SUSv3] | readdir_r(GLIBC_2.2) [SUSv3] |
| readlink(GLIBC_2.2) [SUSv3] | readv(GLIBC_2.2) [SUSv3] | rename(GLIBC_2.2) [SUSv3] | rmdir(GLIBC_2.2) [SUSv3] |
| sbrk(GLIBC_2.2) [SUSv2] | sched_get_priority_max(GLIBC_2.2) [SUSv3] | sched_get_priority_min(GLIBC_2.2) [SUSv3] | sched_getparam(GLIBC_2.2) [SUSv3] |
| sched_getscheduler(GLIBC_2.2) [SUSv3] | sched_rr_get_interval(GLIBC_2.2) [SUSv3] | sched_setparam(GLIBC_2.2) [SUSv3] | sched_setscheduler(GLIBC_2.2) [SUSv3] |
| sched_yield(GLIBC_2.2) [SUSv3] | select(GLIBC_2.2) [SUSv3] | setcontext(GLIBC_2.2) [SUSv3] | setegid(GLIBC_2.2) [SUSv3] |
| seteuid(GLIBC_2.2) [SUSv3] | setgid(GLIBC_2.2) [SUSv3] | setitimer(GLIBC_2.2) [SUSv3] | setpgid(GLIBC_2.2) [SUSv3] |
| setpgrp(GLIBC_2.2) [SUSv3] | setpriority(GLIBC_2.2) [SUSv3] | setregid(GLIBC_2.2) [SUSv3] | setreuid(GLIBC_2.2) [SUSv3] |
| setrlimit(GLIBC_2.2) [SUSv3] | setrlimit64(GLIBC_2.2) [LFS] | setsid(GLIBC_2.2) [SUSv3] | setuid(GLIBC_2.2) [SUSv3] |
| sleep(GLIBC_2.2) [SUSv3] | statvfs(GLIBC_2.2) [SUSv3] | stime(GLIBC_2.2) [LSB] | symlink(GLIBC_2.2) [SUSv3] |
| sync(GLIBC_2.2) [SUSv3] | sysconf(GLIBC_2.2) [SUSv3] | time(GLIBC_2.2) [SUSv3] | times(GLIBC_2.2) [SUSv3] |
| truncate(GLIBC_2.2) [SUSv3] | ulimit(GLIBC_2.2) [SUSv3] | umask(GLIBC_2.2) [SUSv3] | uname(GLIBC_2.2) [SUSv3] |
| unlink(GLIBC_2.2) [LSB] | utime(GLIBC_2.2) [SUSv3] | utimes(GLIBC_2.2) [SUSv3] | vfork(GLIBC_2.2) [SUSv3] |
| wait(GLIBC_2.2) [SUSv3] | wait4(GLIBC_2.2) [LSB] | waitpid(GLIBC_2.2) [LSB] | write(GLIBC_2.2) [SUSv3] |

| | | | |
|---------------------------|--|--|--|
| writev(GLIBC_2.2) [SUSv3] | | | |
|---------------------------|--|--|--|

36

11.2.3 Standard I/O

37

11.2.3.1 Interfaces for Standard I/O

38

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

39

40

41

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

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

42

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

43

Referenced Specification(s)

44

[H]

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

45

46

47

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

48

[2]. ISO POSIX (2003)

49

[3]. SUSv2

50

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

52

53

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

| | | | | |
|-----------------------|----------------------|-----------------------|--|--|
| stderr(GLIBC_2.2) [1] | stdin(GLIBC_2.2) [1] | stdout(GLIBC_2.2) [1] | | |
|-----------------------|----------------------|-----------------------|--|--|

54

Referenced Specification(s)

55

[1]. ISO POSIX (2003)

56

| | | | | |
|---------------------------|--------------------------|---------------------------|--|--|
| stderr(GLIBC_2.2) [SUSv3] | stdin(GLIBC_2.2) [SUSv3] | stdout(GLIBC_2.2) [SUSv3] | | |
|---------------------------|--------------------------|---------------------------|--|--|

57

11.2.4 Signal Handling

58

11.2.4.1 Interfaces for Signal Handling

59

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

60

61

Table 11-6 libc - Signal Handling Function Interfaces

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

63

Referenced Specification(s)

64

65

~~[1]~~

| | | | |
|---|---|--|---|
| __libc_current_sigrtmax(GLIBC_2.2) [LSB] | __libc_current_sigrtmin(GLIBC_2.2) [LSB] | __sigsetjmp(GLIBC_2.2) [LSB] | __sysv_signal(GLIBC_2.2) [LSB] |
| bsd_signal(GLIBC_2.2) [SUSv3] | psignal(GLIBC_2.2) [LSB] | raise(GLIBC_2.2) [SUSv3] | sigaction(GLIBC_2.2) [SUSv3] |
| sigaddset(GLIBC_2.2) [SUSv3] | sigaltstack(GLIBC_2.2) [SUSv3] | sigandset(GLIBC_2.2) [LSB] | sigdelset(GLIBC_2.2) [SUSv3] |
| sigemptyset(GLIBC_2.2) [SUSv3] | sigfillset(GLIBC_2.2) [SUSv3] | sighold(GLIBC_2.2) [SUSv3] | sigignore(GLIBC_2.2) [SUSv3] |
| siginterrupt(GLIBC_2.2) [SUSv3] | sigisemtpyset(GLIBC_2.2) [LSB] | sigismember(GLIBC_2.2) [SUSv3] | siglongjmp(GLIBC_2.2) [SUSv3] |
| signal(GLIBC_2.2) [SUSv3] | sigorset(GLIBC_2.2) [LSB] | sigpause(GLIBC_2.2) [SUSv3] | sigpending(GLIBC_2.2) [SUSv3] |
| sigprocmask(GLIBC_2.2) [SUSv3] | sigqueue(GLIBC_2.2) [SUSv3] | sigrelse(GLIBC_2.2) [SUSv3] | sigreturn(GLIBC_2.2) [LSB] |
| sigset(GLIBC_2.2) [SUSv3] | sigsuspend(GLIBC_2.2) [SUSv3] | sigtimedwait(GLIBC_2.2) [SUSv3] | sigwait(GLIBC_2.2) [SUSv3] |
| sigwaitinfo(GLIBC_2.2) [SUSv3] | | | |

66

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

67

68

69

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

70

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

71

72

73

Table 11-7 libc - Signal Handling Data Interfaces

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

74

75

Referenced Specification(s)

76

~~[1]. this specification~~

77

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

11.2.5 Localization Functions

78

11.2.5.1 Interfaces for Localization Functions

79

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

80

81

82

Table 11-8 libc - Localization Functions Function Interfaces

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

83

84

Referenced Specification(s)

85

~~[1].~~

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

86

87

88

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

89

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

90

91

92

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

93

Table 11-9 libc - Localization Functions Data Interfaces

| | | | | |
|--|--|--|--|--|
| _nl_msg_cat_cntr(GLIBC_2.2) [1] | | | | |
|--|--|--|--|--|

94

95

Referenced Specification(s)

96

~~[1]. this specification~~

| | | | |
|--|--|--|--|
| _nl_msg_cat_cntr(GLIBC_2.2) [LSB] | | | |
|--|--|--|--|

97

11.2.6 Socket Interface

11.2.6.1 Interfaces for Socket Interface

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

Table 11-10 libc - Socket Interface Function Interfaces

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

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

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

| | | | |
|---------------------------|-------------------------------|--|--|
| socket(GLIBC_2.2) [SUSv3] | socketpair(GLIBC_2.2) [SUSv3] | | |
|---------------------------|-------------------------------|--|--|

11.2.7 Wide Characters

11.2.7.1 Interfaces for Wide Characters

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

Table 11-11 libc - Wide Characters Function Interfaces

| | | | | |
|---|---------------------------------------|--|---|---------------------------------------|
| <code>__westod_int</code> <code>ernal(GLIBC_2.2)</code> [1] | <code>mbsinit(GLIBC_2.2)</code> [2] | <code>vwscanf(GLIBC_2.2)</code> [1] | <code>wcsnlen(GLIBC_2.2)</code> [1] | <code>westoumax(GLIBC_2.2)</code> [2] |
| <code>__westof_int</code> <code>ernal(GLIBC_2.2)</code> [1] | <code>mbsrtowes(GLIBC_2.2)</code> [1] | <code>wepepy(GLIBC_2.2)</code> [1] | <code>wcsnrtoombs(GLIBC_2.2)</code> [1] | <code>westouq(GLIBC_2.2)</code> [1] |
| <code>__westol_int</code> <code>ernal(GLIBC_2.2)</code> [1] | <code>mbsrtowes(GLIBC_2.2)</code> [2] | <code>wepnepy(GLIBC_2.2)</code> [1] | <code>wespbrk(GLIBC_2.2)</code> [2] | <code>weswes(GLIBC_2.2)</code> [2] |
| <code>__westold_int</code> <code>ernal(GLIBC_2.2)</code> [1] | <code>mbstowes(GLIBC_2.2)</code> [2] | <code>wertomb(GLIBC_2.2)</code> [2] | <code>wesrchr(GLIBC_2.2)</code> [2] | <code>weswidth(GLIBC_2.2)</code> [2] |
| <code>__westoul_int</code> <code>ernal(GLIBC_2.2)</code> [1] | <code>mbtowe(GLIBC_2.2)</code> [2] | <code>wescaseemp(GLIBC_2.2)</code> [1] | <code>wesrtombs(GLIBC_2.2)</code> [2] | <code>wesxfrm(GLIBC_2.2)</code> [2] |
| <code>btowe(GLIBC_2.2)</code> [2] | <code>putwe(GLIBC_2.2)</code> [2] | <code>wecat(GLIBC_2.2)</code> [2] | <code>wesspn(GLIBC_2.2)</code> [2] | <code>wetob(GLIBC_2.2)</code> [2] |
| <code>fgetwe(GLIBC_2.2)</code> [2] | <code>putwchar(GLIBC_2.2)</code> [2] | <code>weschr(GLIBC_2.2)</code> [2] | <code>wesstr(GLIBC_2.2)</code> [2] | <code>wetomb(GLIBC_2.2)</code> [2] |
| <code>fgetws(GLIBC_2.2)</code> [2] | <code>swprintf(GLIBC_2.2)</code> [2] | <code>wesemp(GLIBC_2.2)</code> [2] | <code>westod(GLIBC_2.2)</code> [2] | <code>wetrans(GLIBC_2.2)</code> [2] |
| <code>fputwe(GLIBC_2.2)</code> [2] | <code>swscanf(GLIBC_2.2)</code> [1] | <code>wescoll(GLIBC_2.2)</code> [2] | <code>westof(GLIBC_2.2)</code> [2] | <code>wetype(GLIBC_2.2)</code> [2] |
| <code>fputws(GLIBC_2.2)</code> [2] | <code>towetrans(GLIBC_2.2)</code> [2] | <code>wesepy(GLIBC_2.2)</code> [2] | <code>westoimax(GLIBC_2.2)</code> [2] | <code>wewidth(GLIBC_2.2)</code> [2] |
| <code>fwide(GLIBC_2.2)</code> [2] | <code>towlower(GLIBC_2.2)</code> [2] | <code>wesespn(GLIBC_2.2)</code> [2] | <code>westok(GLIBC_2.2)</code> [2] | <code>wmemchr(GLIBC_2.2)</code> [2] |
| <code>fwprintf(GLIBC_2.2)</code> [2] | <code>towupper(GLIBC_2.2)</code> [2] | <code>wesdup(GLIBC_2.2)</code> [1] | <code>westol(GLIBC_2.2)</code> [2] | <code>wmememp(GLIBC_2.2)</code> [2] |
| <code>fwscanf(GLIBC_2.2)</code> [1] | <code>ungetwe(GLIBC_2.2)</code> [2] | <code>wesftime(GLIBC_2.2)</code> [2] | <code>westold(GLIBC_2.2)</code> [2] | <code>wmemepy(GLIBC_2.2)</code> [2] |
| <code>getwe(GLIBC_2.2)</code> [2] | <code>vwprintf(GLIBC_2.2)</code> [2] | <code>weslen(GLIBC_2.2)</code> [2] | <code>westoll(GLIBC_2.2)</code> [2] | <code>wmemmove(GLIBC_2.2)</code> [2] |

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

113

114

115

116

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

| | | | |
|--------------------------------------|-------------------------------------|-------------------------------------|--------------------------------------|
| __wcstod_internal (GLIBC_2.2) [LSB] | __wcstof_internal (GLIBC_2.2) [LSB] | __wcstol_internal (GLIBC_2.2) [LSB] | __wcstold_internal (GLIBC_2.2) [LSB] |
| __wcstoul_internal (GLIBC_2.2) [LSB] | btowc (GLIBC_2.2) [SUSv3] | fgetwc (GLIBC_2.2) [SUSv3] | fgetws (GLIBC_2.2) [SUSv3] |
| fputwc (GLIBC_2.2) [SUSv3] | fputws (GLIBC_2.2) [SUSv3] | fwide (GLIBC_2.2) [SUSv3] | fwprintf (GLIBC_2.2) [SUSv3] |
| fwscanf (GLIBC_2.2) [LSB] | getwc (GLIBC_2.2) [SUSv3] | getwchar (GLIBC_2.2) [SUSv3] | mblen (GLIBC_2.2) [SUSv3] |
| mbrlen (GLIBC_2.2) [SUSv3] | mbrtowc (GLIBC_2.2) [SUSv3] | mbsinit (GLIBC_2.2) [SUSv3] | mbsrtowcs (GLIBC_2.2) [LSB] |
| mbsrtowcs (GLIBC_2.2) [SUSv3] | mbstowcs (GLIBC_2.2) [SUSv3] | mbtowc (GLIBC_2.2) [SUSv3] | putwc (GLIBC_2.2) [SUSv3] |
| putwchar (GLIBC_2.2) [SUSv3] | swprintf (GLIBC_2.2) [SUSv3] | swscanf (GLIBC_2.2) [LSB] | towctrans (GLIBC_2.2) [SUSv3] |
| tolower (GLIBC_2.2) [SUSv3] | toupper (GLIBC_2.2) [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_2.2) [LSB] | wcpncpy (GLIBC_2.2) [LSB] | wcrtomb (GLIBC_2.2) [SUSv3] |
| wcscasecmp (GLIBC_2.2) [LSB] | wcscat (GLIBC_2.2) [SUSv3] | wcschr (GLIBC_2.2) [SUSv3] | wcscmp (GLIBC_2.2) [SUSv3] |
| wscoll (GLIBC_2.2) [SUSv3] | wcscpy (GLIBC_2.2) [SUSv3] | wcscspn (GLIBC_2.2) [SUSv3] | wcsdup (GLIBC_2.2) [LSB] |
| wcsftime (GLIBC_2.2) [SUSv3] | wcslen (GLIBC_2.2) [SUSv3] | wcscasecmp (GLIBC_2.2) [LSB] | wcscat (GLIBC_2.2) [SUSv3] |

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

117

11.2.8 String Functions

118

11.2.8.1 Interfaces for String Functions

119

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.

120

121

122

Table 11-12 libc - String Functions Function Interfaces

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

| | | | | |
|---|--|---|---------------------------------------|---------------------------------------|
| <code>nal(GLIBC_2.2)</code> [1] | <code>BC_2.2)</code> [2] | <code>_2.2)</code> [2] | <code>C_2.2)</code> [2] | <code>E_2.2)</code> [2] |
| <code>__strtok_r(GLIBC_2.2)</code> [1] | <code>memcpy(GLIBC_2.2)</code> [2] | <code>strespn(GLIBC_2.2)</code> [2] | <code>strptime(GLIBC_2.2)</code> [1] | <code>strtoumax(GLIBC_2.2)</code> [2] |
| <code>__strtol_internal(GLIBC_2.2)</code> [1] | <code>memmove(GLIBC_2.2)</code> [2] | <code>strdup(GLIBC_2.2)</code> [2] | <code>strchr(GLIBC_2.2)</code> [2] | <code>strtouq(GLIBC_2.2)</code> [1] |
| <code>__strtold_internal(GLIBC_2.2)</code> [1] | <code>memrchr(GLIBC_2.2)</code> [1] | <code>strerror(GLIBC_2.2)</code> [2] | <code>strsep(GLIBC_2.2)</code> [1] | <code>strxfrm(GLIBC_2.2)</code> [2] |
| <code>__strtoll_internal(GLIBC_2.2)</code> [1] | <code>memset(GLIBC_2.2)</code> [2] | <code>strerror_r(GLIBC_2.2)</code> [1] | <code>strsignal(GLIBC_2.2)</code> [1] | <code>swab(GLIBC_2.2)</code> [2] |
| <code>__strtoul_internal(GLIBC_2.2)</code> [1] | <code>rindex(GLIBC_2.2)</code> [2] | <code>strfmon(GLIBC_2.2)</code> [2] | <code>strspn(GLIBC_2.2)</code> [2] | |
| <code>__strtoull_internal(GLIBC_2.2)</code> [1] | <code>stpcpy(GLIBC_2.2)</code> [1] | <code>strtime(GLIBC_2.2)</code> [2] | <code>strstr(GLIBC_2.2)</code> [2] | |
| <code>bcmp(GLIBC_2.2)</code> [2] | <code>stpncpy(GLIBC_2.2)</code> [1] | <code>strlen(GLIBC_2.2)</code> [2] | <code>strtof(GLIBC_2.2)</code> [2] | |
| <code>bcopy(GLIBC_2.2)</code> [2] | <code>strcasecmp(GLIBC_2.2)</code> [2] | <code>strncasecmp(GLIBC_2.2)</code> [2] | <code>strtoimax(GLIBC_2.2)</code> [2] | |

123

124

125

126

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

| | | | |
|--|--|--|---|
| <code>__memcpy(GLIBC_C_2.2)</code> [LSB] | <code>__rawmemchr(GLIBC_2.2)</code> [LSB] | <code>__stpcpy(GLIBC_2.2)</code> [LSB] | <code>__strdup(GLIBC_2.2)</code> [LSB] |
| <code>__strtod_internal(GLIBC_2.2)</code> [LSB] | <code>__strtof_internal(GLIBC_2.2)</code> [LSB] | <code>__strtok_r(GLIBC_2.2)</code> [LSB] | <code>__strtol_internal(GLIBC_2.2)</code> [LSB] |
| <code>__strtold_internal(GLIBC_2.2)</code> [LSB] | <code>__strtoll_internal(GLIBC_2.2)</code> [LSB] | <code>__strtoul_internal(GLIBC_2.2)</code> [LSB] | <code>__strtoull_internal(GLIBC_2.2)</code> [LSB] |
| <code>bcmp(GLIBC_2.2)</code> [SUSv3] | <code>bcopy(GLIBC_2.2)</code> [SUSv3] | <code>bzero(GLIBC_2.2)</code> [SUSv3] | <code>ffs(GLIBC_2.2)</code> [SUSv3] |
| <code>index(GLIBC_2.2)</code> [SUSv3] | <code>memcpy(GLIBC_2.2)</code> [SUSv3] | <code>memchr(GLIBC_2.2)</code> [SUSv3] | <code>memcmp(GLIBC_2.2)</code> [SUSv3] |
| <code>memcpy(GLIBC_2.2)</code> [SUSv3] | <code>memmove(GLIBC_2.2)</code> [SUSv3] | <code>memrchr(GLIBC_2.2)</code> [LSB] | <code>memset(GLIBC_2.2)</code> [SUSv3] |
| <code>rindex(GLIBC_2.2)</code> [SUSv3] | <code>stpcpy(GLIBC_2.2)</code> [LSB] | <code>stpncpy(GLIBC_2.2)</code> [LSB] | <code>strcasecmp(GLIBC_2.2)</code> [SUSv3] |
| <code>strcasestr(GLIBC_2.2)</code> [SUSv3] | <code>strcat(GLIBC_2.2)</code> [SUSv3] | <code>strchr(GLIBC_2.2)</code> [SUSv3] | <code>strcmp(GLIBC_2.2)</code> [SUSv3] |

| | | | |
|-----------------------------|--------------------------------|------------------------------|------------------------------|
| 2.2) [LSB] | [SUSv3] | [SUSv3] |) [SUSv3] |
| strcoll(GLIBC_2.2) [SUSv3] | strcpy(GLIBC_2.2) [SUSv3] | strcspn(GLIBC_2.2) [SUSv3] | strdup(GLIBC_2.2) [SUSv3] |
| strerror(GLIBC_2.2) [SUSv3] | strerror_r(GLIBC_2.2) [LSB] | strfmon(GLIBC_2.2) [SUSv3] | strftime(GLIBC_2.2) [SUSv3] |
| strlen(GLIBC_2.2) [SUSv3] | strncasecmp(GLIBC_2.2) [SUSv3] | strncat(GLIBC_2.2) [SUSv3] | strncmp(GLIBC_2.2) [SUSv3] |
| strncpy(GLIBC_2.2) [SUSv3] | strndup(GLIBC_2.2) [LSB] | strnlen(GLIBC_2.2) [LSB] | strpbrk(GLIBC_2.2) [SUSv3] |
| strptime(GLIBC_2.2) [LSB] | strchr(GLIBC_2.2) [SUSv3] | strsep(GLIBC_2.2) [LSB] | strsignal(GLIBC_2.2) [LSB] |
| strspn(GLIBC_2.2) [SUSv3] | strstr(GLIBC_2.2) [SUSv3] | strtof(GLIBC_2.2) [SUSv3] | strtoimax(GLIBC_2.2) [SUSv3] |
| strtok(GLIBC_2.2) [SUSv3] | strtok_r(GLIBC_2.2) [SUSv3] | strtold(GLIBC_2.2) [SUSv3] | strtoll(GLIBC_2.2) [SUSv3] |
| strtoq(GLIBC_2.2) [LSB] | strtoull(GLIBC_2.2) [SUSv3] | strtoumax(GLIBC_2.2) [SUSv3] | strtouq(GLIBC_2.2) [LSB] |
| strxfrm(GLIBC_2.2) [SUSv3] | swab(GLIBC_2.2) [SUSv3] | | |

127

11.2.9 IPC Functions

128

11.2.9.1 Interfaces for IPC Functions

129

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.

130

131

132

Table 11-13 libc - IPC Functions Function Interfaces

| | | | | |
|-----------------------|-----------------------|-----------------------|-----------------------|--|
| ftok(GLIBC_2.2) [1] | msgrev(GLIBC_2.2) [1] | semget(GLIBC_2.2) [1] | shmetl(GLIBC_2.2) [1] | |
| msgctl(GLIBC_2.2) [1] | msgsnd(GLIBC_2.2) [1] | semop(GLIBC_2.2) [1] | shmdt(GLIBC_2.2) [1] | |
| msgget(GLIBC_2.2) [1] | semctl(GLIBC_2.2) [1] | shmat(GLIBC_2.2) [1] | shmget(GLIBC_2.2) [1] | |

133

134

Referenced Specification(s)

135

[1]. ISO POSIX (2003)

| | | | |
|---------------------------|---------------------------|---------------------------|---------------------------|
| ftok(GLIBC_2.2) [SUSv3] | msgctl(GLIBC_2.2) [SUSv3] | msgget(GLIBC_2.2) [SUSv3] | msgrcv(GLIBC_2.2) [SUSv3] |
| msgsnd(GLIBC_2.2) [SUSv3] | semctl(GLIBC_2.2) [SUSv3] | semget(GLIBC_2.2) [SUSv3] | semop(GLIBC_2.2) [SUSv3] |
| shmat(GLIBC_2.2) [SUSv3] | shmetl(GLIBC_2.2) [SUSv3] | shmdt(GLIBC_2.2) [SUSv3] | shmget(GLIBC_2.2) [SUSv3] |

136

11.2.10 Regular Expressions

11.2.10.1 Interfaces for Regular Expressions

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

Table 11-14 libc - Regular Expressions Function Interfaces

| | | | | |
|--------------------------------------|---------------------------------------|--|--------------------------------------|--|
| <code>regcomp</code> (GLIBC_2.2) [1] | <code>regerror</code> (GLIBC_2.2) [1] | <code>regexec</code> (GLIBC_2.3.4) [2] | <code>regfree</code> (GLIBC_2.2) [1] | |
|--------------------------------------|---------------------------------------|--|--------------------------------------|--|

Referenced Specification(s)

[1]. ISO POSIX (2003)

[2]. this specification

| | | | | |
|--|---|--|--|--|
| <code>regcomp</code> (GLIBC_2.2) [SUSv3] | <code>regerror</code> (GLIBC_2.2) [SUSv3] | <code>regexec</code> (GLIBC_2.3.4) [LSB] | <code>regfree</code> (GLIBC_2.2) [SUSv3] | |
|--|---|--|--|--|

11.2.11 Character Type Functions

11.2.11.1 Interfaces for Character Type Functions

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

Table 11-15 libc - Character Type Functions Function Interfaces

| | | | | |
|--|--------------------------------------|---------------------------------------|--|--------------------------------------|
| <code>_ctype_get_mb_cur_max</code> (GLIBC_2.2) [1] | <code>isdigit</code> (GLIBC_2.2) [2] | <code>iswalnum</code> (GLIBC_2.2) [2] | <code>iswlower</code> (GLIBC_2.2) [2] | <code>toascii</code> (GLIBC_2.2) [2] |
| <code>_tolower</code> (GLIBC_2.2) [2] | <code>isgraph</code> (GLIBC_2.2) [2] | <code>iswalpha</code> (GLIBC_2.2) [2] | <code>iswprint</code> (GLIBC_2.2) [2] | <code>tolower</code> (GLIBC_2.2) [2] |
| <code>_toupper</code> (GLIBC_2.2) [2] | <code>islower</code> (GLIBC_2.2) [2] | <code>iswblank</code> (GLIBC_2.2) [2] | <code>iswpunct</code> (GLIBC_2.2) [2] | <code>toupper</code> (GLIBC_2.2) [2] |
| <code>isalnum</code> (GLIBC_2.2) [2] | <code>isprint</code> (GLIBC_2.2) [2] | <code>iswendl</code> (GLIBC_2.2) [2] | <code>iswspace</code> (GLIBC_2.2) [2] | |
| <code>isalpha</code> (GLIBC_2.2) [2] | <code>ispunct</code> (GLIBC_2.2) [2] | <code>iswctype</code> (GLIBC_2.2) [2] | <code>iswupper</code> (GLIBC_2.2) [2] | |
| <code>isascii</code> (GLIBC_2.2) [2] | <code>isspace</code> (GLIBC_2.2) [2] | <code>iswdigit</code> (GLIBC_2.2) [2] | <code>iswxdigit</code> (GLIBC_2.2) [2] | |
| <code>isendl</code> (GLIBC_2.2) [2] | <code>isupper</code> (GLIBC_2.2) [2] | <code>iswgraph</code> (GLIBC_2.2) [2] | <code>isxdigit</code> (GLIBC_2.2) [2] | |

Referenced Specification(s)

[1]. this specification

[2]. ISO POSIX (2003)

| | | | |
|--|--|---|--|
| <code>__ctype_get_mb_cur_max(GLIBC_2.2)</code> [LSB] | <code>_tolower(GLIBC_2.2)</code> [SUSv3] | <code>_toupper(GLIBC_2.2)</code> [SUSv3] | <code>isalnum(GLIBC_2.2)</code> [SUSv3] |
| <code>isalpha(GLIBC_2.2)</code> [SUSv3] | <code>isascii(GLIBC_2.2)</code> [SUSv3] | <code>iscntrl(GLIBC_2.2)</code> [SUSv3] | <code>isdigit(GLIBC_2.2)</code> [SUSv3] |
| <code>isgraph(GLIBC_2.2)</code> [SUSv3] | <code>islower(GLIBC_2.2)</code> [SUSv3] | <code>isprint(GLIBC_2.2)</code> [SUSv3] | <code>ispunct(GLIBC_2.2)</code> [SUSv3] |
| <code>isspace(GLIBC_2.2)</code> [SUSv3] | <code>isupper(GLIBC_2.2)</code> [SUSv3] | <code>iswalnum(GLIBC_2.2)</code> [SUSv3] | <code>iswalpha(GLIBC_2.2)</code> [SUSv3] |
| <code>iswblank(GLIBC_2.2)</code> [SUSv3] | <code>iswcntrl(GLIBC_2.2)</code> [SUSv3] | <code>iswctype(GLIBC_2.2)</code> [SUSv3] | <code>iswdigit(GLIBC_2.2)</code> [SUSv3] |
| <code>iswgraph(GLIBC_2.2)</code> [SUSv3] | <code>iswlower(GLIBC_2.2)</code> [SUSv3] | <code>iswprint(GLIBC_2.2)</code> [SUSv3] | <code>iswpunct(GLIBC_2.2)</code> [SUSv3] |
| <code>iswspace(GLIBC_2.2)</code> [SUSv3] | <code>iswupper(GLIBC_2.2)</code> [SUSv3] | <code>iswxdigit(GLIBC_2.2)</code> [SUSv3] | <code>isxdigit(GLIBC_2.2)</code> [SUSv3] |
| <code>toascii(GLIBC_2.2)</code> [SUSv3] | <code>tolower(GLIBC_2.2)</code> [SUSv3] | <code>toupper(GLIBC_2.2)</code> [SUSv3] | |

156

11.2.12 Time Manipulation

157

11.2.12.1 Interfaces for Time Manipulation

158

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.

159

160

161

Table 11-16 libc - Time Manipulation Function Interfaces

| | | | | |
|---------------------------------------|--------------------------------------|---------------------------------------|---|------------------------------------|
| <code>adjtime(GLIBC_2.2)</code> [1] | <code>etime(GLIBC_2.2)</code> [2] | <code>gmtime(GLIBC_2.2)</code> [2] | <code>localtime_r(GLIBC_2.2)</code> [2] | <code>ualarm(GLIBC_2.2)</code> [2] |
| <code>asctime(GLIBC_2.2)</code> [2] | <code>etime_r(GLIBC_2.2)</code> [2] | <code>gmtime_r(GLIBC_2.2)</code> [2] | <code>mktime(GLIBC_2.2)</code> [2] | |
| <code>asctime_r(GLIBC_2.2)</code> [2] | <code>difftime(GLIBC_2.2)</code> [2] | <code>localtime(GLIBC_2.2)</code> [2] | <code>tzset(GLIBC_2.2)</code> [2] | |

162

163

Referenced Specification(s)

164

[1]

| | | | |
|---|---|---|--|
| <code>adjtime(GLIBC_2.2)</code> [LSB] | <code>asctime(GLIBC_2.2)</code> [SUSv3] | <code>asctime_r(GLIBC_2.2)</code> [SUSv3] | <code>ctime(GLIBC_2.2)</code> [SUSv3] |
| <code>ctime_r(GLIBC_2.2)</code> [SUSv3] | <code>difftime(GLIBC_2.2)</code> [SUSv3] | <code>gmtime(GLIBC_2.2)</code> [SUSv3] | <code>gmtime_r(GLIBC_2.2)</code> [SUSv3] |
| <code>localtime(GLIBC_2.2)</code> [SUSv3] | <code>localtime_r(GLIBC_2.2)</code> [SUSv3] | <code>mktime(GLIBC_2.2)</code> [SUSv3] | <code>tzset(GLIBC_2.2)</code> [SUSv3] |
| <code>ualarm(GLIBC_2.2)</code> [SUSv3] | | | |

165

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

168 ~~[2]. ISO POSIX (2003)~~

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

172 **Table 11-17 libc - Time Manipulation Data Interfaces**

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

173 *Referenced Specification(s)*

174 ~~[1]. this specification~~

175 ~~[2]. ISO POSIX (2003)~~

| | | | |
|--|--|--------------------------------------|--|
| __daylight(GLIBC_2.2) [LSB] | __timezone(GLIBC_2.2) [LSB] | __tzname(GLIBC_2.2) [LSB] | daylight(GLIBC_2.2) [SUSv3] |
| timezone(GLIBC_2.2) [SUSv3] | tzname(GLIBC_2.2) [SUSv3] | | |

177

11.2.13 Terminal Interface Functions

11.2.13.1 Interfaces for Terminal Interface Functions

178 An LSB conforming implementation shall provide the architecture specific functions
 179 for Terminal Interface Functions specified in Table 11-18, with the full mandatory
 180 functionality as described in the referenced underlying specification.
 181

182 **Table 11-18 libc - Terminal Interface Functions Function Interfaces**

| | | | | |
|---------------------------------------|---------------------------------------|-----------------------------------|-------------------------------------|---------------------------------------|
| efgetispeed(GLIBC_2.2) [1] | efsetispeed(GLIBC_2.2) [1] | tedrain(GLIBC_2.2) [1] | tegetattr(GLIBC_2.2) [1] | tesendbreak(GLIBC_2.2) [1] |
| efgetospeed(GLIBC_2.2) [1] | efsetospeed(GLIBC_2.2) [1] | tcflow(GLIBC_2.2) [1] | tegetpgrp(GLIBC_2.2) [1] | tcsetattr(GLIBC_2.2) [1] |
| efmakeraw(GLIBC_2.2) [2] | efsetspeed(GLIBC_2.2) [2] | tcflush(GLIBC_2.2) [1] | tegetsid(GLIBC_2.2) [1] | tcsetpgrp(GLIBC_2.2) [1] |

183

184 *Referenced Specification(s)*

185 ~~[1]. ISO POSIX (2003)~~

186 ~~[2]. this specification~~

| | | | |
|---|---|---------------------------------------|---|
| cfgetispeed(GLIBC_2.2) [SUSv3] | cfgetospeed(GLIBC_2.2) [SUSv3] | cfmakeraw(GLIBC_2.2) [LSB] | cfsetispeed(GLIBC_2.2) [SUSv3] |
| cfsetospeed(GLIBC_2.2) [SUSv3] | cfsetspeed(GLIBC_2.2) [SUSv3] | tcdrain(GLIBC_2.2) [SUSv3] | tcflow(GLIBC_2.2) [SUSv3] |

| | | | |
|--------------------------------|------------------------------|------------------------------|-----------------------------|
| C_2.2) [SUSv3] | _2.2) [LSB] | 2) [SUSv3] |) [SUSv3] |
| tcflush(GLIBC_2.2) [SUSv3] | tcgetattr(GLIBC_2.2) [SUSv3] | tcgetpgrp(GLIBC_2.2) [SUSv3] | tcgetsid(GLIBC_2.2) [SUSv3] |
| tcsendbreak(GLIBC_2.2) [SUSv3] | tcsetattr(GLIBC_2.2) [SUSv3] | tcsetpgrp(GLIBC_2.2) [SUSv3] | |

187

11.2.14 System Database Interface

188

11.2.14.1 Interfaces for System Database Interface

189

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.

190

191

192

Table 11-19 libc - System Database Interface Function Interfaces

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

193

194

Referenced Specification(s)

195

[1]. ISO POSIX (2003)

196

[2]. this specification

197

[3]. SUSv2

| | | | |
|-----------------------------|--------------------------------|-----------------------------|-------------------------------|
| endgrent(GLIBC_2.2) [SUSv3] | endprotoent(GLIBC_2.2) [SUSv3] | endpwent(GLIBC_2.2) [SUSv3] | endservent(GLIBC_2.2) [SUSv3] |
| endutent(GLIBC_2.2) [SUSv3] | endutxent(GLIBC_2.2) [SUSv3] | getgrent(GLIBC_2.2) [SUSv3] | getgrgid(GLIBC_2.2) [SUSv3] |

| | | | |
|----------------------------------|----------------------------------|-----------------------------------|-------------------------------------|
| 2.2) [SUSv2] | _.2) [SUSv3] | .2) [SUSv3] | .2) [SUSv3] |
| getgrgid_r(GLIBC_2.2) [SUSv3] | getgrnam(GLIBC_2.2) [SUSv3] | getgrnam_r(GLIBC_2.2) [SUSv3] | getgrouplist(GLIBC_2.2.4) [LSB] |
| gethostbyaddr(GLIBC_2.2) [SUSv3] | gethostbyname(GLIBC_2.2) [SUSv3] | getprotobyname(GLIBC_2.2) [SUSv3] | getprotobynumber(GLIBC_2.2) [SUSv3] |
| getprotoent(GLIBC_2.2) [SUSv3] | getpwent(GLIBC_2.2) [SUSv3] | getpwnam(GLIBC_2.2) [SUSv3] | getpwnam_r(GLIBC_2.2) [SUSv3] |
| getpwuid(GLIBC_2.2) [SUSv3] | getpwuid_r(GLIBC_2.2) [SUSv3] | getservbyname(GLIBC_2.2) [SUSv3] | getservbyport(GLIBC_2.2) [SUSv3] |
| getservent(GLIBC_2.2) [SUSv3] | getutent(GLIBC_2.2) [LSB] | getutent_r(GLIBC_2.2) [LSB] | getutxent(GLIBC_2.2) [SUSv3] |
| getutxid(GLIBC_2.2) [SUSv3] | getutxline(GLIBC_2.2) [SUSv3] | pututxline(GLIBC_2.2) [SUSv3] | setgrent(GLIBC_2.2) [SUSv3] |
| setgroups(GLIBC_2.2) [LSB] | setprotoent(GLIBC_2.2) [SUSv3] | setpwent(GLIBC_2.2) [SUSv3] | setservent(GLIBC_2.2) [SUSv3] |
| setutent(GLIBC_2.2) [LSB] | setutxent(GLIBC_2.2) [SUSv3] | utmpname(GLIBC_2.2) [LSB] | |

198

11.2.15 Language Support

199

11.2.15.1 Interfaces for Language Support

200

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.

201

202

203

Table 11-20 libc - Language Support Function Interfaces

| | | | | |
|--|--|--|--|--|
| __libc_start_main(GLIBC_2.2)[1] | | | | |
|--|--|--|--|--|

204

205

Referenced Specification(s)

206

~~[1].this specification~~

207

| | | | |
|---|--|--|--|
| __libc_start_main(GLIBC_2.2) [LSB] | | | |
|---|--|--|--|

11.2.16 Large File Support

208

11.2.16.1 Interfaces for Large File Support

209

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.

210

211

212

Table 11-21 libc - Large File Support Function Interfaces

| | | | | |
|----------------------------|-------------------------|--------------------------|------------------------|--------------------------|
| __fxstat64(GLIB | fopen64(GLIB | ftello64(GLIB | mkstemp64(G | tmpfile64(GLI |
|----------------------------|-------------------------|--------------------------|------------------------|--------------------------|

| | | | | |
|--|--|---|---------------------------------------|--|
| <code>IBC_2.2)</code> [1] | <code>C_2.2)</code> [2] | <code>C_2.2)</code> [2] | <code>LIBC_2.2)</code> [2] | <code>BC_2.2)</code> [2] |
| <code>__lxstat64(GLIBC_2.2)</code> [1] | <code>freopen64(GLIBC_2.2)</code> [2] | <code>ftruncate64(GLIBC_2.2)</code> [2] | <code>mmap64(GLIBC_2.2)</code> [2] | <code>truncate64(GLIBC_2.2)</code> [2] |
| <code>__xstat64(GLIBC_2.2)</code> [1] | <code>fseeko64(GLIBC_2.2)</code> [2] | <code>ftw64(GLIBC_2.2)</code> [2] | <code>nftw64(GLIBC_2.3.3)</code> [2] | |
| <code>creat64(GLIBC_2.2)</code> [2] | <code>fsetpos64(GLIBC_2.2)</code> [2] | <code>getrlimit64(GLIBC_2.2)</code> [2] | <code>readdir64(GLIBC_2.2)</code> [2] | |
| <code>fgetpos64(GLIBC_2.2)</code> [2] | <code>fstatvfs64(GLIBC_2.2)</code> [2] | <code>lockf64(GLIBC_2.2)</code> [2] | <code>statvfs64(GLIBC_2.2)</code> [2] | |

213

214

Referenced Specification(s)

215

[1]. this specification

216

[2]. Large File Support

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

217

11.2.17 Standard Library

218

11.2.17.1 Interfaces for Standard Library

219

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.

220

221

Table 11-22 libc - Standard Library Function Interfaces

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

222

| | | | | |
|--|--|--|--|---|
| <code>__errno_location(GLIBC_2.2)</code> [2] | <code>evvt(GLIBC_2.2)</code> [1] | <code>globfree(GLIBC_2.2)</code> [1] | <code>malloc(GLIBC_2.2)</code> [1] | <code>strtod(GLIBC_2.2)</code> [1] |
| <code>__fpending(GLIBC_2.2)</code> [2] | <code>erand48(GLIBC_2.2)</code> [1] | <code>globfree64(GLIBC_2.2)</code> [2] | <code>memmem(GLIBC_2.2)</code> [2] | <code>strtol(GLIBC_2.2)</code> [1] |
| <code>__getpagesize(GLIBC_2.2)</code> [2] | <code>err(GLIBC_2.2)</code> [2] | <code>grantpt(GLIBC_2.2)</code> [1] | <code>mkstemp(GLIBC_2.2)</code> [1] | <code>strtoul(GLIBC_2.2)</code> [1] |
| <code>__isinf(GLIBC_2.2)</code> [2] | <code>error(GLIBC_2.2)</code> [2] | <code>hereate(GLIBC_2.2)</code> [1] | <code>mktemp(GLIBC_2.2)</code> [1] | <code>swapcontext(GLIBC_2.2)</code> [1] |
| <code>__isinf(GLIBC_2.2)</code> [2] | <code>errx(GLIBC_2.2)</code> [2] | <code>hdestroy(GLIBC_2.2)</code> [1] | <code>mrnd48(GLIBC_2.2)</code> [1] | <code>syslog(GLIBC_2.2)</code> [1] |
| <code>__isnfl(GLIBC_2.2)</code> [2] | <code>fevt(GLIBC_2.2)</code> [1] | <code>hsearch(GLIBC_2.2)</code> [1] | <code>nftw(GLIBC_2.3.3)</code> [1] | <code>system(GLIBC_2.2)</code> [2] |
| <code>__isnan(GLIBC_2.2)</code> [2] | <code>fmtmsg(GLIBC_2.2)</code> [1] | <code>htonl(GLIBC_2.2)</code> [1] | <code>nrnd48(GLIBC_2.2)</code> [1] | <code>tdelete(GLIBC_2.2)</code> [1] |
| <code>__isnan(GLIBC_2.2)</code> [2] | <code>fnmatch(GLIBC_2.2.3)</code> [1] | <code>htons(GLIBC_2.2)</code> [1] | <code>ntohl(GLIBC_2.2)</code> [1] | <code>tfind(GLIBC_2.2)</code> [1] |
| <code>__isnan(GLIBC_2.2)</code> [2] | <code>fpathconf(GLIBC_2.2)</code> [1] | <code>imaxabs(GLIBC_2.2)</code> [1] | <code>ntohs(GLIBC_2.2)</code> [1] | <code>tmpfile(GLIBC_2.2)</code> [1] |
| <code>__sysconf(GLIBC_2.2)</code> [2] | <code>free(GLIBC_2.2)</code> [1] | <code>imaxdiv(GLIBC_2.2)</code> [1] | <code>openlog(GLIBC_2.2)</code> [1] | <code>tmpnam(GLIBC_2.2)</code> [1] |
| <code>_exit(GLIBC_2.2)</code> [1] | <code>freeaddrinfo(GLIBC_2.2)</code> [1] | <code>inet_addr(GLIBC_2.2)</code> [1] | <code>perror(GLIBC_2.2)</code> [1] | <code>tsearch(GLIBC_2.2)</code> [1] |
| <code>_longjmp(GLIBC_2.2)</code> [1] | <code>ftrylockfile(GLIBC_2.2)</code> [1] | <code>inet_ntoa(GLIBC_2.2)</code> [1] | <code>posix_memalign(GLIBC_2.2)</code> [1] | <code>ttyname(GLIBC_2.2)</code> [1] |
| <code>_setjmp(GLIBC_2.2)</code> [1] | <code>ftw(GLIBC_2.2)</code> [1] | <code>inet_ntop(GLIBC_2.2)</code> [1] | <code>posix_openpt(GLIBC_2.2.1)</code> [1] | <code>ttyname_r(GLIBC_2.2)</code> [1] |
| <code>a64l(GLIBC_2.2)</code> [1] | <code>funlockfile(GLIBC_2.2)</code> [1] | <code>inet_pton(GLIBC_2.2)</code> [1] | <code>ptsname(GLIBC_2.2)</code> [1] | <code>twalk(GLIBC_2.2)</code> [1] |
| <code>abort(GLIBC_2.2)</code> [1] | <code>gai_strerror(GLIBC_2.2)</code> [1] | <code>initstate(GLIBC_2.2)</code> [1] | <code>putenv(GLIBC_2.2)</code> [1] | <code>unlockpt(GLIBC_2.2)</code> [1] |
| <code>abs(GLIBC_2.2)</code> [1] | <code>gevt(GLIBC_2.2)</code> [1] | <code>insque(GLIBC_2.2)</code> [1] | <code>qsort(GLIBC_2.2)</code> [1] | <code>unsetenv(GLIBC_2.2)</code> [1] |
| <code>atof(GLIBC_2.2)</code> [1] | <code>getaddrinfo(GLIBC_2.2)</code> [1] | <code>isatty(GLIBC_2.2)</code> [1] | <code>rand(GLIBC_2.2)</code> [1] | <code>usleep(GLIBC_2.2)</code> [1] |
| <code>atoi(GLIBC_2.2)</code> [1] | <code>getwd(GLIBC_2.2)</code> [1] | <code>isblank(GLIBC_2.2)</code> [1] | <code>rand_r(GLIBC_2.2)</code> [1] | <code>verrx(GLIBC_2.2)</code> [2] |

| | | | | |
|-------------------------|---------------------------------|------------------------|----------------------------|--------------------------|
| atol(GLIBC_2.2) [1] | getdate(GLIBC_2.2) [1] | jrand48(GLIBC_2.2) [1] | random(GLIBC_2.2) [1] | vfscanf(GLIBC_2.2) [2] |
| atoll(GLIBC_2.2) [1] | getenv(GLIBC_2.2) [1] | l64a(GLIBC_2.2) [1] | realloc(GLIBC_2.2) [1] | vscanf(GLIBC_2.2) [2] |
| basename(GLIBC_2.2) [1] | getlogin(GLIBC_2.2) [1] | labs(GLIBC_2.2) [1] | realpath(GLIBC_2.3) [1] | vsscanf(GLIBC_2.2) [2] |
| bsearch(GLIBC_2.2) [1] | getlogin_r(GLIBC_2.2) [1] | lcong48(GLIBC_2.2) [1] | remque(GLIBC_2.2) [1] | vsyslog(GLIBC_2.2) [2] |
| calloc(GLIBC_2.2) [1] | getnameinfo(GLIBC_2.2) [1] | ldiv(GLIBC_2.2) [1] | seed48(GLIBC_2.2) [1] | warn(GLIBC_2.2) [2] |
| closelog(GLIBC_2.2) [1] | getopt(GLIBC_2.2) [2] | lfind(GLIBC_2.2) [1] | setenv(GLIBC_2.2) [1] | warnx(GLIBC_2.2) [2] |
| confstr(GLIBC_2.2) [1] | getopt_long(GLIBC_2.2) [2] | llabs(GLIBC_2.2) [1] | sethostname(GLIBC_2.2) [2] | wordexp(GLIBC_2.2.2) [1] |
| cuserid(GLIBC_2.2) [3] | getopt_long_only(GLIBC_2.2) [2] | lldiv(GLIBC_2.2) [1] | setlogmask(GLIBC_2.2) [1] | wordfree(GLIBC_2.2) [1] |
| daemon(GLIBC_2.2) [2] | getsubopt(GLIBC_2.2) [1] | longjmp(GLIBC_2.2) [1] | setstate(GLIBC_2.2) [1] | |

223

224

225

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

| | | | |
|-----------------------------|--------------------------------|-------------------------------|-----------------------------------|
| _Exit(GLIBC_2.2) [SUSv3] | __assert_fail(GLIBC_2.2) [LSB] | __cxa_atexit(GLIBC_2.2) [LSB] | __errno_location(GLIBC_2.2) [LSB] |
| __fpending(GLIBC_2.2) [LSB] | __getpagesize(GLIBC_2.2) [LSB] | __isinf(GLIBC_2.2) [LSB] | __isinf(GLIBC_2.2) [LSB] |
| __isnfl(GLIBC_2.2) [LSB] | __isnan(GLIBC_2.2) [LSB] | __isnanf(GLIBC_2.2) [LSB] | __isnanl(GLIBC_2.2) [LSB] |
| __sysconf(GLIBC_2.2) [LSB] | _exit(GLIBC_2.2) [SUSv3] | _longjmp(GLIBC_2.2) [SUSv3] | _setjmp(GLIBC_2.2) [SUSv3] |
| a64l(GLIBC_2.2) [SUSv3] | abort(GLIBC_2.2) [SUSv3] | abs(GLIBC_2.2) [SUSv3] | atof(GLIBC_2.2) [SUSv3] |
| atoi(GLIBC_2.2) [SUSv3] | atol(GLIBC_2.2) [SUSv3] | atoll(GLIBC_2.2) [SUSv3] | basename(GLIBC_2.2) [SUSv3] |
| bsearch(GLIBC_2.2) [SUSv3] | calloc(GLIBC_2.2) [SUSv3] | closelog(GLIBC_2.2) [SUSv3] | confstr(GLIBC_2.2) [SUSv3] |
| cuserid(GLIBC_2.2) [SUSv2] | daemon(GLIBC_2.2) [LSB] | dirname(GLIBC_2.2) [SUSv3] | div(GLIBC_2.2) [SUSv3] |
| drand48(GLIBC_2.2) [SUSv3] | ecvt(GLIBC_2.2) [SUSv3] | erand48(GLIBC_2.2) [SUSv3] | err(GLIBC_2.2) [LSB] |

| | | | |
|-----------------------------------|-----------------------------------|-----------------------------------|---------------------------------|
| error(GLIBC_2.2) [LSB] | errx(GLIBC_2.2) [LSB] | fcvt(GLIBC_2.2) [SUSv3] | fmtmsg(GLIBC_2.2) [SUSv3] |
| fnmatch(GLIBC_2.3) [SUSv3] | fpathconf(GLIBC_2.2) [SUSv3] | free(GLIBC_2.2) [SUSv3] | freeaddrinfo(GLIBC_2.2) [SUSv3] |
| ftrylockfile(GLIBC_2.2) [SUSv3] | ftw(GLIBC_2.2) [SUSv3] | funlockfile(GLIBC_2.2) [SUSv3] | gai_strerror(GLIBC_2.2) [SUSv3] |
| gcvt(GLIBC_2.2) [SUSv3] | getaddrinfo(GLIBC_2.2) [SUSv3] | getcwd(GLIBC_2.2) [SUSv3] | getdate(GLIBC_2.2) [SUSv3] |
| getenv(GLIBC_2.2) [SUSv3] | getlogin(GLIBC_2.2) [SUSv3] | getlogin_r(GLIBC_2.2) [SUSv3] | getnameinfo(GLIBC_2.2) [SUSv3] |
| getopt(GLIBC_2.2) [LSB] | getopt_long(GLIBC_2.2) [LSB] | getopt_long_only(GLIBC_2.2) [LSB] | getsubopt(GLIBC_2.2) [SUSv3] |
| gettimeofday(GLIBC_2.2) [SUSv3] | glob(GLIBC_2.2) [SUSv3] | glob64(GLIBC_2.2) [LSB] | globfree(GLIBC_2.2) [SUSv3] |
| globfree64(GLIBC_2.2) [LSB] | grantpt(GLIBC_2.2) [SUSv3] | hcreate(GLIBC_2.2) [SUSv3] | hdestroy(GLIBC_2.2) [SUSv3] |
| hsearch(GLIBC_2.2) [SUSv3] | htonl(GLIBC_2.2) [SUSv3] | htons(GLIBC_2.2) [SUSv3] | imaxabs(GLIBC_2.2) [SUSv3] |
| imaxdiv(GLIBC_2.2) [SUSv3] | inet_addr(GLIBC_2.2) [SUSv3] | inet_ntoa(GLIBC_2.2) [SUSv3] | inet_ntop(GLIBC_2.2) [SUSv3] |
| inet_pton(GLIBC_2.2) [SUSv3] | initstate(GLIBC_2.2) [SUSv3] | insque(GLIBC_2.2) [SUSv3] | isatty(GLIBC_2.2) [SUSv3] |
| isblank(GLIBC_2.2) [SUSv3] | jrand48(GLIBC_2.2) [SUSv3] | l64a(GLIBC_2.2) [SUSv3] | labs(GLIBC_2.2) [SUSv3] |
| lcong48(GLIBC_2.2) [SUSv3] | ldiv(GLIBC_2.2) [SUSv3] | lfind(GLIBC_2.2) [SUSv3] | llabs(GLIBC_2.2) [SUSv3] |
| lldiv(GLIBC_2.2) [SUSv3] | longjmp(GLIBC_2.2) [SUSv3] | lrand48(GLIBC_2.2) [SUSv3] | lsearch(GLIBC_2.2) [SUSv3] |
| makecontext(GLIBC_2.2) [SUSv3] | malloc(GLIBC_2.2) [SUSv3] | memmem(GLIBC_2.2) [LSB] | mkstemp(GLIBC_2.2) [SUSv3] |
| mktemp(GLIBC_2.2) [SUSv3] | mrnd48(GLIBC_2.2) [SUSv3] | nftw(GLIBC_2.3.3) [SUSv3] | nrnd48(GLIBC_2.2) [SUSv3] |
| ntohl(GLIBC_2.2) [SUSv3] | ntohs(GLIBC_2.2) [SUSv3] | openlog(GLIBC_2.2) [SUSv3] | perror(GLIBC_2.2) [SUSv3] |
| posix_memalign(GLIBC_2.2) [SUSv3] | posix_openpt(GLIBC_2.2.1) [SUSv3] | ptsname(GLIBC_2.2) [SUSv3] | putenv(GLIBC_2.2) [SUSv3] |
| qsort(GLIBC_2.2) [SUSv3] | rand(GLIBC_2.2) [SUSv3] | rand_r(GLIBC_2.2) [SUSv3] | random(GLIBC_2.2) [SUSv3] |
| realloc(GLIBC_2.2) [SUSv3] | realpath(GLIBC_2.3) [SUSv3] | remque(GLIBC_2.2) [SUSv3] | seed48(GLIBC_2.2) [SUSv3] |

| | | | |
|-----------------------------|------------------------------|--------------------------------|------------------------------|
| setenv(GLIBC_2.2) [SUSv3] | sethostname(GLIBC_2.2) [LSB] | setlogmask(GLIBC_2.2) [SUSv3] | setstate(GLIBC_2.2) [SUSv3] |
| srand(GLIBC_2.2) [SUSv3] | srand48(GLIBC_2.2) [SUSv3] | srandom(GLIBC_2.2) [SUSv3] | strtod(GLIBC_2.2) [SUSv3] |
| strtol(GLIBC_2.2) [SUSv3] | strtoul(GLIBC_2.2) [SUSv3] | swapcontext(GLIBC_2.2) [SUSv3] | syslog(GLIBC_2.2) [SUSv3] |
| system(GLIBC_2.2) [LSB] | tdelete(GLIBC_2.2) [SUSv3] | tfind(GLIBC_2.2) [SUSv3] | tmpfile(GLIBC_2.2) [SUSv3] |
| tmpnam(GLIBC_2.2) [SUSv3] | tsearch(GLIBC_2.2) [SUSv3] | ttyname(GLIBC_2.2) [SUSv3] | ttyname_r(GLIBC_2.2) [SUSv3] |
| twalk(GLIBC_2.2) [SUSv3] | unlockpt(GLIBC_2.2) [SUSv3] | unsetenv(GLIBC_2.2) [SUSv3] | usleep(GLIBC_2.2) [SUSv3] |
| verrx(GLIBC_2.2) [LSB] | vfscanf(GLIBC_2.2) [LSB] | vscanf(GLIBC_2.2) [LSB] | vsscanf(GLIBC_2.2) [LSB] |
| vsyslog(GLIBC_2.2) [LSB] | warn(GLIBC_2.2) [LSB] | warnx(GLIBC_2.2) [LSB] | wordexp(GLIBC_2.2.2) [SUSv3] |
| wordfree(GLIBC_2.2) [SUSv3] | | | |

226

227

228

229

230

231

232

233

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

~~[2]. this specification~~

~~[3]. SUSv2~~

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.

234

Table 11-23 libc - Standard Library Data Interfaces

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

235

236

237

238

Referenced Specification(s)

~~[1]. this specification~~

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

| | | | |
|---|--------------------------------------|--|---------------------------------------|
| __environ(GLIBC_2.2) [LSB] | _environ(GLIBC_2.2) [LSB] | _sys_errlist(GLIBC_2.3) [LSB] | environ(GLIBC_2.2) [SUSv3] |
| getdate_err(GLIBC_2.2) [SUSv3] | optarg(GLIBC_2.2) [SUSv3] | opterr(GLIBC_2.2) [SUSv3] | optind(GLIBC_2.2) [SUSv3] |
| optopt(GLIBC_2.2) [SUSv3] | | | |

239

11.3 Data Definitions for libc

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

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

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

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

11.3.1 arpa/inet.h

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

11.3.2 assert.h

```
265
266 extern void __assert_fail(const char *, const char *, unsigned int,
267                          const char *);
```

11.3.3 ctype.h

```
268
269 extern int _tolower(int);
270 extern int _toupper(int);
271 extern int isalnum(int);
272 extern int isalpha(int);
273 extern int isascii(int);
274 extern int iscntrl(int);
275 extern int isdigit(int);
276 extern int isgraph(int);
277 extern int islower(int);
278 extern int isprint(int);
279 extern int ispunct(int);
280 extern int isspace(int);
281 extern int isupper(int);
282 extern int isxdigit(int);
283 extern int toascii(int);
284 extern int tolower(int);
285 extern int toupper(int);
286 extern int isblank(int);
```

```

287 extern const unsigned short **__ctype_b_loc(void);
288 extern const int32_t **__ctype_toupper_loc(void);
289 extern const int32_t **__ctype_tolower_loc(void);

```

11.3.4 dirent.h

```

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

```

11.3.5 err.h

```

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

```

11.3.6 errno.h

```

305
306 #define EDEADLOCK      EDEADLK
307
308 extern int *__errno_location(void);

```

11.3.27 fcntl.h

```

309
310 #define F_GETLK64      5
311 #define F_SETLK64      6
312 #define F_SETLKW64     7

```

11.3.3 inttypes.h

```

313
314 typedef long int intmax_t;
315 typedef unsigned long int uintmax_t;
316 typedef unsigned long int uintptr_t;
317 typedef unsigned long int uint64_t;

```

11.3.4 limits.h

```

318
319 #define LONG_MAX      0x7FFFFFFFFFFFFFFFL
320 #define ULONG_MAX     0xFFFFFFFFFFFFFFFFUL
321
322 #define CHAR_MAX     SCHAR_MAX
323 #define CHAR_MIN     SCHAR_MIN
324
325 #define PTHREAD_STACK_MIN 196608

```

11.3.5 setjmp.h

```

326
327 extern int lockf64(int, int, off64_t);
328 extern int fcntl(int, int, ...);

```

11.3.8 fmtmsg.h

```

329
330 extern int fmtmsg(long int, const char *, int, const char *, const char
331 *,
332                    const char *);

```

11.3.9 fnmatch.h

```

333
334 extern int fnmatch(const char *, const char *, int);

```

11.3.10 ftw.h

```

335
336 extern int ftw(const char *, __ftw_func_t, int);
337 extern int ftw64(const char *, __ftw64_func_t, int);
338 extern int nftw(const char *, __nftw_func_t, int, int);
339 extern int nftw64(const char *, __nftw64_func_t, int, int);

```

11.3.11 getopt.h

```

340
341 typedef long int __jmp_buf[70] __attribute__((aligned(16)));

```

11.3.6 signal.h

```

342
343 #define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int)) 4)
344
345 #define SI_PAD_SIZE ((SI_MAX_SIZE/sizeof(int)) 4)
346
347 struct sigaction
348 {
349     union
350     {
351         sighandler_t __sa_handler;
352         void (*__sa_sigaction) (int, siginfo_t *, void *);
353     }
354     __sigaction_handler;
355     unsigned long int sa_flags;
356     sigset_t sa_mask;
357 }
358 ;
359 #define MINSIGSTKSZ 131027
360 #define SIGSTKSZ 262144
361
362 struct ia64_fpreg
363 {
364     union
365     {
366         unsigned long int bits[2];
367         long double __dummy;
368     }
369 };
370
371 ;
372 extern int getopt_long(int, char *const, const char *,
373                       const struct option *, int *);
374 extern int getopt_long_only(int, char *const, const char *,
375                             const struct option *, int *);

```


11.3.12 glob.h

```

376
377 extern int glob(const char *, int,
378                int (*__errfunc) (const char *p1, int p2)
379                , glob_t *);
380 extern int glob64(const char *, int,
381                 int (*__errfunc) (const char *p1, int p2)
382                 , glob64_t *);
383 extern void globfree(glob_t *);
384 extern void globfree64(glob64_t *);

```

11.3.13 grp.h

```

385
386 extern void endgrent(void);
387 extern struct group *getgrent(void);
388 extern struct group *getgrgid(gid_t);
389 extern struct group *getgrnam(char *);
390 extern int initgroups(const char *, gid_t);
391 extern void setgrent(void);
392 extern int setgroups(size_t, const gid_t *);
393 extern int getgrgid_r(gid_t, struct group *, char *, size_t,
394                      struct group **);
395 extern int getgrnam_r(const char *, struct group *, char *, size_t,
396                      struct group **);
397 extern int getgrouplist(const char *, gid_t, gid_t *, int *);

```

11.3.14 iconv.h

```

398
399 extern size_t iconv(iconv_t, char **, size_t *, char **, size_t *);
400 extern int iconv_close(iconv_t);
401 extern iconv_t iconv_open(char *, char *);

```

11.3.15 inttypes.h

```

402
403 typedef long int intmax_t;
404 typedef unsigned long int uintmax_t;
405 typedef unsigned long int uintptr_t;
406 typedef unsigned long int uint64_t;
407
408 struct sigcontext
409 {
410     unsigned long int sc_flags;
411     unsigned long int sc_nat;
412     stack_t sc_stack;
413     unsigned long int sc_ip;
414     unsigned long int sc_efm;
415     unsigned long int sc_um;
416     unsigned long int sc_ar_rsc;
417     unsigned long int sc_ar_bsp;
418     unsigned long int sc_ar_rnat;
419     unsigned long int sc_ar_ccv;
420     unsigned long int sc_ar_unat;
421     unsigned long int sc_ar_fpsr;
422     unsigned long int sc_ar_pfs;
423     unsigned long int sc_ar_le;
424     unsigned long int sc_pr;
425     unsigned long int sc_br[8];
426     unsigned long int sc_gr[32];
427     struct ia64_fpreg sc_fr[128];

```

```

428     — unsigned long int se_rbs_base;
429     — unsigned long int se_loadrs;
430     — unsigned long int se_ar25;
431     — unsigned long int se_ar26;
432     — unsigned long int se_rsvd[12];
433     — unsigned long int se_mack;
434     }
435     ;

```

11.3.7 stddef.h

```

436
437     typedef long int ptrdiff_t;
438     typedef unsigned long int size_t;

```

11.3.8 stdio.h

```

439
440     #define __IO_FILE_SIZE 216

```

11.3.9 sys/ioctl.h

```

441
442     #define TIOCGWINSZ 0x5413
443     #define FIONREAD 0x541B
444     #define TIOCNOTTY 0x5422

```

11.3.10 sys/ipc.h

```

445
446     struct ipc_perm
447     {
448     — key_t __key;
449     — uid_t uid;
450     — gid_t gid;
451     — uid_t cuid;
452     — uid_t egid;
453     — mode_t mode;
454     — unsigned short __seq;
455     — unsigned short __pad1;
456     — unsigned long int __unused1;
457     — unsigned long int __unused2;
458     }
459     ;

```

11.3.11 sys/mman.h

```

460
461     #define MCL_CURRENT 1
462     #define MCL_FUTURE 2

```

11.3.12 sys/msg.h

```

463
464     struct msgid_ds
465     {
466     — struct ipc_perm msg_perm;
467     — time_t msg_stime;
468     — time_t msg_rtime;
469     — time_t msg_ctime;
470     — unsigned long int __msg_cbytes;
471     — unsigned long int msg_qnum;

```

```

472 —unsigned long int msg_qbytes;
473 —pid_t msg_lspid;
474 —pid_t msg_lrpid;
475 —unsigned long int __unused1;
476 —unsigned long int __unused2;
477 };
478 };

```

11.3.13 sys/sem.h

```

479 extern intmax_t strtoumax(const char *, char **, int);
480 extern uintmax_t strtoumax(const char *, char **, int);
481 extern intmax_t wcstoumax(const wchar_t *, wchar_t * *, int);
482 extern uintmax_t wcstoumax(const wchar_t *, wchar_t * *, int);
483 extern intmax_t imaxabs(intmax_t);
484 extern imaxdiv_t imaxdiv(intmax_t, intmax_t);

```

11.3.16 langinfo.h

```

485
486 extern char *nl_langinfo(nl_item);

```

11.3.17 libgen.h

```

487
488 extern char *basename(const char *);
489 extern char *dirname(char *);

```

11.3.18 libintl.h

```

490
491 extern char *bindtextdomain(const char *, const char *);
492 extern char *dcgettext(const char *, const char *, int);
493 extern char *dgettext(const char *, const char *);
494 extern char *gettext(const char *);
495 extern char *textdomain(const char *);
496 extern char *bind_textdomain_codeset(const char *, const char *);
497 extern char *dcngettext(const char *, const char *, const char *,
498                        unsigned long int, int);
499 extern char *dngettext(const char *, const char *, const char *,
500                        unsigned long int);
501 extern char *ngettext(const char *, const char *, unsigned long int);

```

11.3.19 limits.h

```

502
503 #define LONG_MAX          0x7FFFFFFFFFFFFFFFL
504 #define ULONG_MAX        0xFFFFFFFFFFFFFFFFUL
505
506 #define CHAR_MAX          SCHAR_MAX
507 #define CHAR_MIN          SCHAR_MIN
508
509 #define PTHREAD_STACK_MIN 196608

```

11.3.20 locale.h

```

510
511 struct semid_ds
512 {
513 —struct ipc_perm sem_perm;
514 —time_t sem_otime;
515 —time_t sem_ctime;
516 —unsigned long int sem_nsems;

```

```

517     — unsigned long int __unused1;
518     — unsigned long int __unused2;
519     }
520     —;

```

11.3.14 sys/shm.h

```

521
522     #define SHMLBA (1024*1024)
523
524     struct shmid_ds
525     {
526     — struct ipc_perm shm_perm;
527     — size_t shm_segsz;
528     — time_t shm_atime;
529     — time_t shm_dtime;
530     — time_t shm_ctime;
531     — pid_t shm_epid;
532     — pid_t shm_lpid;
533     — unsigned long int shm_nattch;
534     — unsigned long int __unused1;
535     — unsigned long int __unused2;
536     }
537     —;

```

11.3.15 sys/socket.h

```

538
539     typedef uint64_t __ss_aligntype;
540
541     #define SO_RCVLOWAT 18
542     #define SO_SNDLOWAT 19
543     #define SO_RCVTIMEO 20
544     #define SO_SNDTIMEO 21

```

11.3.16 sys/stat.h

```

545
546     #define __STAT_VER 1
547
548     struct stat
549     {
550     — dev_t st_dev;
551     — ino_t st_ino;
552     — nlink_t st_nlink;
553     — mode_t st_mode;
554     — uid_t st_uid;
555     — gid_t st_gid;
556     — unsigned int pad0;
557     — dev_t st_rdev;
558     — off_t st_size;
559     — struct timespec st_atim;
560     — struct timespec st_mtim;
561     — struct timespec st_ctim;
562     — blksize_t st_blksize;
563     — blkent_t st_blocks;
564     — unsigned long int __unused[3];
565     }
566     —;
567     struct stat64
568     {
569     — dev_t st_dev;
570     — ino64_t st_ino;

```

```

571 —nlink_t st_nlink;
572 —mode_t st_mode;
573 —uid_t st_uid;
574 —gid_t st_gid;
575 —unsigned int pad0;
576 —dev_t st_rdev;
577 —off_t st_size;
578 —struct timespec st_atim;
579 —struct timespec st_mtim;
580 —struct timespec st_ctim;
581 —blksize_t st_blksize;
582 —blkcnt64_t st_blocks;
583 —unsigned long int __unused[3];
584 }
585 —;

```

11.3.17 sys/statvfs.h

```

586 extern struct lconv *localeconv(void);
587 extern char *setlocale(int, const char *);
588 extern locale_t uselocale(locale_t);
589 extern void freelocale(locale_t);
590 extern locale_t duplocale(locale_t);
591 extern locale_t newlocale(int, const char *, locale_t);

```

11.3.21 monetary.h

```

592
593 extern ssize_t strfmon(char *, size_t, const char *, ...);

```

11.3.22 net/if.h

```

594
595 extern void if_freenameindex(struct if_nameindex *);
596 extern char *if_indextoname(unsigned int, char *);
597 extern struct if_nameindex *if_nameindex(void);
598 extern unsigned int if_nametoindex(const char *);

```

11.3.23 netdb.h

```

599
600 extern void endprotoent(void);
601 extern void endservent(void);
602 extern void freeaddrinfo(struct addrinfo *);
603 extern const char *gai_strerror(int);
604 extern int getaddrinfo(const char *, const char *, const struct addrinfo
605 *,
606                       struct addrinfo **);
607 extern struct hostent *gethostbyaddr(const void *, socklen_t, int);
608 extern struct hostent *gethostbyname(const char *);
609 extern struct protoent *getprotobyname(const char *);
610 extern struct protoent *getprotobynumber(int);
611 extern struct protoent *getprotoent(void);
612 extern struct servent *getservbyname(const char *, const char *);
613 extern struct servent *getservbyport(int, const char *);
614 extern struct servent *getservent(void);
615 extern void setprotoent(int);
616 extern void setservent(int);
617 extern int *__h_errno_location(void);

```

11.3.24 netinet/in.h

```

618
619 extern int bindresvport(int, struct sockaddr_in *);

```

11.3.25 netinet/ip.h

```

620
621 /*
622  * This header is architecture neutral
623  * Please refer to the generic specification for details
624  */

```

11.3.26 netinet/tcp.h

```

625
626 /*
627  * This header is architecture neutral
628  * Please refer to the generic specification for details
629  */

```

11.3.27 netinet/udp.h

```

630
631 struct statvfs
632 {
633     unsigned long int f_bsize;
634     unsigned long int f_frsize;
635     fsblkcnt64_t f_blocks;
636     fsblkcnt64_t f_bfree;
637     fsblkcnt64_t f_bavail;
638     fsfilcnt64_t f_files;
639     fsfilcnt64_t f_ffree;
640     fsfilcnt64_t f_favail;
641     unsigned long int f_fsid;
642     unsigned long int f_flag;
643     unsigned long int f_namemax;
644     unsigned int __f_spare[6];
645 }
646 };
647 struct statvfs64
648 {
649     unsigned long int f_bsize;
650     unsigned long int f_frsize;
651     fsblkcnt64_t f_blocks;
652     fsblkcnt64_t f_bfree;
653     fsblkcnt64_t f_bavail;
654     fsfilcnt64_t f_files;
655     fsfilcnt64_t f_ffree;
656     fsfilcnt64_t f_favail;
657     unsigned long int f_fsid;
658     unsigned long int f_flag;
659     unsigned long int f_namemax;
660     unsigned int __f_spare[6];
661 }
662 };

```

11.3.18 sys/types.h

```

663
664 typedef long int int64_t;
665
666 typedef int64_t ssize_t;
667
668 #define __FDSET_LONGS 16

```

11.3.19 termios.h

```

669
670     #define OLCUC 0000002
671     #define ONLCR 0000004
672     #define XCASE 0000004
673     #define NLDLY 0000400
674     #define CR1 0001000
675     #define IUCLC 0001000
676     #define CR2 0002000
677     #define CR3 0003000
678     #define CRDLY 0003000
679     #define TAB1 0004000
680     #define TAB2 0010000
681     #define TAB3 0014000
682     #define TABDLY 0014000
683     #define BS1 0020000
684     #define BSDLY 0020000
685     #define VT1 0040000
686     #define VTDLY 0040000
687     #define FF1 0100000
688     #define FFDLY 0100000
689
690     #define VSUSP 10
691     #define VEOL 11
692     #define VREPRINT 12
693     #define VDISCARD 13
694     #define VWERASE 14
695     #define VEOL2 16
696     #define VMIN 6
697     #define VSWTC 7
698     #define VSTART 8
699     #define VSTOP 9
700
701     #define IXON 0002000
702     #define IXOFF 0010000
703
704     #define CS6 0000020
705     #define CS7 0000040
706     #define CS8 0000060
707     #define CSIZE 0000060
708     #define CSTOPB 0000100
709     #define CREAD 0000200
710     #define PARENB 0000400
711     #define PARODD 0001000
712     #define HUPCL 0002000
713     #define CLOCAL 0004000
714     #define VTIME 5
715
716     #define ISIG 0000001
717     #define ICANON 0000002
718     #define ECHOE 0000020
719     #define ECHOK 0000040
720     #define ECHONL 0000100
721     #define NOFLSH 0000200
722     #define TOSTOP 0000400
723     #define ECHOCTL 0001000
724     #define ECHOPRT 0002000
725     #define ECHOKE 0004000
726     #define FLUSHO 0010000
727     #define PENDIN 0040000
728     #define IEXTEN 0100000

```

11.3.20 ucontext.h

```

729 /*
730  * This header is architecture neutral
731  * Please refer to the generic specification for details
732  */

```

11.3.28 nl_types.h

```

733
734 extern int catclose(nl_catd);
735 extern char *catgets(nl_catd, int, int, const char *);
736 extern nl_catd catopen(const char *, int);

```

11.3.29 poll.h

```

737
738 extern int poll(struct pollfd *, nfd_t, int);

```

11.3.30 pty.h

```

739
740 extern int openpty(int *, int *, char *, struct termios *,
741                  struct winsize *);
742 extern int forkpty(int *, char *, struct termios *, struct winsize *);

```

11.3.31 pwd.h

```

743
744 extern void endpwent(void);
745 extern struct passwd *getpwent(void);
746 extern struct passwd *getpwnam(char *);
747 extern struct passwd *getpwuid(uid_t);
748 extern void setpwent(void);
749 extern int getpwnam_r(char *, struct passwd *, char *, size_t,
750                      struct passwd **);
751 extern int getpwuid_r(uid_t, struct passwd *, char *, size_t,
752                      struct passwd **);

```

11.3.32 regex.h

```

753
754 extern int regcomp(regex_t *, const char *, int);
755 extern size_t regerror(int, const regex_t *, char *, size_t);
756 extern int regexec(const regex_t *, const char *, size_t, regmatch_t,
757                   int);
758 extern void regfree(regex_t *);

```

11.3.33 rpc/auth.h

```

759
760 extern struct AUTH *authnone_create(void);
761 extern int key_decryptsession(char *, union des_block *);
762 extern bool_t xdr_opaque_auth(XDR *, struct opaque_auth *);

```

11.3.34 rpc/clnt.h

```

763
764 extern struct CLIENT *clnt_create(const char *, const u_long, const
765 u_long,
766                                 const char *);
767 extern void clnt_pcreateerror(const char *);
768 extern void clnt_perrno(enum clnt_stat);

```



```

769 extern void clnt_perror(struct CLIENT *, const char *);
770 extern char *clnt_spcreateerror(const char *);
771 extern char *clnt_sperrno(enum clnt_stat);
772 extern char *clnt_spperror(struct CLIENT *, const char *);

```

11.3.35 rpc/pmap_clnt.h

```

773
774 extern u_short pmap_getport(struct sockaddr_in *, const u_long,
775                             const u_long, u_int);
776 extern bool_t pmap_set(const u_long, const u_long, int, u_short);
777 extern bool_t pmap_unset(u_long, u_long);

```

11.3.36 rpc/rpc_msg.h

```

778
779 #define _SC_GRP0_OFFSET ((char *) &((struct sigcontext *) 0) -> sc_gr[0])
780 -(char *) 0)
781
782 typedef struct sigcontext mccontext_t;
783
784 typedef struct ucontext
785 {
786     union
787     {
788         mccontext_t _mc;
789         struct
790         {
791             unsigned long int _pad[_SC_GRP0_OFFSET / 8];
792             struct ucontext *_link;
793         }
794     } _uc;
795 };
796 __u;
797 };
798 ucontext_t;

```

11.3.21 unistd.h

```

799
800 typedef long int intptr_t;

```

11.3.22 utmp.h

```

801 extern bool_t xdr_callhdr(XDR *, struct rpc_msg *);

```

11.3.37 rpc/svc.h

```

802
803 extern void svc_getreqset(fd_set *);
804 extern bool_t svc_register(SVCXPRT *, rpcprog_t, rpcvers_t,
805                             __dispatch_fn_t, rpcprot_t);
806
807 extern void svc_run(void);
808 extern bool_t svc_sendreply(SVCXPRT *, xdrproc_t, caddr_t);
809 extern void svcerr_auth(SVCXPRT *, enum auth_stat);
810 extern void svcerr_decode(SVCXPRT *);
811 extern void svcerr_noproc(SVCXPRT *);
812 extern void svcerr_noprogram(SVCXPRT *);
813 extern void svcerr_progvers(SVCXPRT *, rpcvers_t, rpcvers_t);
814 extern void svcerr_systemerr(SVCXPRT *);
815 extern void svcerr_weakauth(SVCXPRT *);
816 extern SVCXPRT *svctcp_create(int, u_int, u_int);
817 extern SVCXPRT *svcdp_create(int);

```

11.3.38 rpc/types.h

```

817
818 /*
819  * This header is architecture neutral
820  * Please refer to the generic specification for details
821  */

```

11.3.39 rpc/xdr.h

```

822
823 extern bool_t xdr_array(XDR *, caddr_t *, u_int *, u_int, u_int,
824                       xdrproc_t);
825 extern bool_t xdr_bool(XDR *, bool_t *);
826 extern bool_t xdr_bytes(XDR *, char **, u_int *, u_int);
827 extern bool_t xdr_char(XDR *, char *);
828 extern bool_t xdr_double(XDR *, double *);
829 extern bool_t xdr_enum(XDR *, enum_t *);
830 extern bool_t xdr_float(XDR *, float *);
831 extern void xdr_free(xdrproc_t, char *);
832 extern bool_t xdr_int(XDR *, int *);
833 extern bool_t xdr_long(XDR *, long int *);
834 extern bool_t xdr_opaque(XDR *, caddr_t, u_int);
835 extern bool_t xdr_pointer(XDR *, char **, u_int, xdrproc_t);
836 extern bool_t xdr_reference(XDR *, caddr_t *, u_int, xdrproc_t);
837 extern bool_t xdr_short(XDR *, short *);
838 extern bool_t xdr_string(XDR *, char **, u_int);
839 extern bool_t xdr_u_char(XDR *, u_char *);
840 extern bool_t xdr_u_int(XDR *, u_int *);
841 extern bool_t xdr_u_long(XDR *, u_long *);
842 extern bool_t xdr_u_short(XDR *, u_short *);
843 extern bool_t xdr_union(XDR *, enum_t *, char *,
844                       const struct xdr_discrim *, xdrproc_t);
845 extern bool_t xdr_vector(XDR *, char *, u_int, u_int, xdrproc_t);
846 extern bool_t xdr_void(void);
847 extern bool_t xdr_wrapstring(XDR *, char **);
848 extern void xdrmem_create(XDR *, caddr_t, u_int, enum xdr_op);
849 extern void xdrrec_create(XDR *, u_int, u_int, caddr_t,
850                          int (*__readit) (char *p1, char *p2, int p3)
851                          , int (*__writeit) (char *p1, char *p2, int
852                          p3)
853                          );
854 extern typedef int bool_t xdrrec_eof(XDR *);

```

11.3.40 sched.h

```

855
856 struct lastlog
857 {
858     time_t ll_time;
859     char ll_line[UT_LINESIZE];
860     char ll_host[UT_HOSTSIZE];
861 }
862 →
863 extern int sched_get_priority_max(int);
864 extern int sched_get_priority_min(int);
865 extern int sched_getparam(pid_t, struct sched_param *);
866 extern int sched_getscheduler(pid_t);
867 extern int sched_rr_get_interval(pid_t, struct timespec *);
868 extern int sched_setparam(pid_t, const struct sched_param *);
869 extern int sched_setscheduler(pid_t, int, const struct sched_param *);
870 extern int sched_yield(void);

```

11.3.41 search.h

```

871
872 struct utmp
873 {
874     short ut_type;
875     pid_t ut_pid;
876     char ut_line[UT_LINESIZE];
877     char ut_id[4];
878     char ut_user[UT_NAMESIZE];
879     char ut_host[UT_HOSTSIZE];
880     struct exit_status ut_exit;
881     long int ut_session;
882     struct timeval ut_tv;
883     int32_t ut_addr_v6[4];
884     char __unused[20];
885 }
886 ;

```

11.3.23 utmpx.h

```

887 extern int hcreate(size_t);
888 extern ENTRY *hsearch(ENTRY, ACTION);
889 extern void insque(void *, void *);
890 extern void *lfind(const void *, const void *, size_t *, size_t,
891     __compar_fn_t);
892 extern void *lsearch(const void *, void *, size_t *, size_t,
893     __compar_fn_t);
894 extern void remque(void *);
895 extern void hdestroy(void);
896 extern void *tdelete(const void *, void **, __compar_fn_t);
897 extern void *tfind(const void *, void *const *, __compar_fn_t);
898 extern void *tsearch(const void *, void **, __compar_fn_t);
899 extern void twalk(const void *, __action_fn_t);

```

11.3.42 setjmp.h

```

900
901 struct utmpx
902 {
903     short ut_type;
904     pid_t ut_pid;
905     char ut_line[UT_LINESIZE];
906     char ut_id[4];
907     char ut_user[UT_NAMESIZE];
908     char ut_host[UT_HOSTSIZE];
909     struct exit_status ut_exit;
910     long int ut_session;
911     struct timeval ut_tv;
912     int32_t ut_addr_v6[4];
913     char __unused[20];
914 }
915 ;
916 typedef long int __jmp_buf[70] __attribute__((aligned(16)));
917
918 extern int __sigsetjmp(jmp_buf, int);
919 extern void longjmp(jmp_buf, int);
920 extern void siglongjmp(sigjmp_buf, int);
921 extern void _longjmp(jmp_buf, int);
922 extern int _setjmp(jmp_buf);

```

11.3.43 signal.h

923

```

924     #define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int))-4)
925
926     #define SI_PAD_SIZE ((SI_MAX_SIZE/sizeof(int))-4)
927
928     struct sigaction {
929         union {
930             sighandler_t _sa_handler;
931             void (*_sa_sigaction) (int, siginfo_t *, void *);
932         } __sigaction_handler;
933         unsigned long int sa_flags;
934         sigset_t sa_mask;
935     };
936
937     #define MINSIGSTKSZ 131027
938     #define SIGSTKSZ 262144
939
940     struct ia64_fpreg {
941         union {
942             unsigned long int bits[2];
943             long double __dummy;
944         } u;
945     };
946
947     struct sigcontext {
948         unsigned long int sc_flags;
949         unsigned long int sc_nat;
950         stack_t sc_stack;
951         unsigned long int sc_ip;
952         unsigned long int sc_cfm;
953         unsigned long int sc_um;
954         unsigned long int sc_ar_rsc;
955         unsigned long int sc_ar_bsp;
956         unsigned long int sc_ar_rnat;
957         unsigned long int sc_ar_ccv;
958         unsigned long int sc_ar_unat;
959         unsigned long int sc_ar_fpsr;
960         unsigned long int sc_ar_pfs;
961         unsigned long int sc_ar_lc;
962         unsigned long int sc_pr;
963         unsigned long int sc_br[8];
964         unsigned long int sc_gr[32];
965         struct ia64_fpreg sc_fr[128];
966         unsigned long int sc_rbs_base;
967         unsigned long int sc_loadrs;
968         unsigned long int sc_ar25;
969         unsigned long int sc_ar26;
970         unsigned long int sc_rsvd[12];
971         unsigned long int sc_mask;
972     };
973     extern int __libc_current_sigrtmax(void);
974     extern int __libc_current_sigrtmin(void);
975     extern sighandler_t __sysv_signal(int, sighandler_t);
976     extern char *const _sys_siglist(void);
977     extern int killpg(pid_t, int);
978     extern void psignal(int, const char *);
979     extern int raise(int);
980     extern int sigaddset(sigset_t *, int);
981     extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
982     extern int sigdelset(sigset_t *, int);
983     extern int sigemptyset(sigset_t *);
984     extern int sigfillset(sigset_t *);
985     extern int sighold(int);
986     extern int sigignore(int);
987     extern int siginterrupt(int, int);

```

```

988 extern int sigisemptyset(const sigset_t *);
989 extern int sigismember(const sigset_t *, int);
990 extern int sigorset(sigset_t *, const sigset_t *, const sigset_t *);
991 extern int sigpending(sigset_t *);
992 extern int sigrelse(int);
993 extern sighandler_t sigset(int, sighandler_t);
994 extern int pthread_kill(pthread_t, int);
995 extern int pthread_sigmask(int, sigset_t *, sigset_t *);
996 extern int sigaction(int, const struct sigaction *, struct sigaction *);
997 extern int sigwait(sigset_t *, int *);
998 extern int kill(pid_t, int);
999 extern int sigaltstack(const struct sigaltstack *, struct sigaltstack
1000 *);
1001 extern sighandler_t signal(int, sighandler_t);
1002 extern int sigpause(int);
1003 extern int sigprocmask(int, const sigset_t *, sigset_t *);
1004 extern int sigreturn(struct sigcontext *);
1005 extern int sigsuspend(const sigset_t *);
1006 extern int sigqueue(pid_t, int, const union sigval);
1007 extern int sigwaitinfo(const sigset_t *, siginfo_t *);
1008 extern int sigtimedwait(const sigset_t *, siginfo_t *,
1009                         const struct timespec *);
1010 extern sighandler_t bsd_signal(int, sighandler_t);

```

11.3.44 stddef.h

```

1011
1012 typedef long int ptrdiff_t;
1013 typedef unsigned long int size_t;

```

11.3.45 stdio.h

```

1014
1015 #define __IO_FILE_SIZE 216
1016
1017 extern char *const _sys_errlist(void);
1018 extern void clearerr(FILE *);
1019 extern int fclose(FILE *);
1020 extern FILE *fdopen(int, const char *);
1021 extern int fflush_unlocked(FILE *);
1022 extern int fileno(FILE *);
1023 extern FILE *fopen(const char *, const char *);
1024 extern int fprintf(FILE *, const char *, ...);
1025 extern int fputc(int, FILE *);
1026 extern FILE *freopen(const char *, const char *, FILE *);
1027 extern FILE *freopen64(const char *, const char *, FILE *);
1028 extern int fscanf(FILE *, const char *, ...);
1029 extern int fseek(FILE *, long int, int);
1030 extern int fseeko(FILE *, off_t, int);
1031 extern int fseeko64(FILE *, loff_t, int);
1032 extern off_t ftello(FILE *);
1033 extern loff_t ftello64(FILE *);
1034 extern int getchar(void);
1035 extern int getchar_unlocked(void);
1036 extern int getw(FILE *);
1037 extern int pclose(FILE *);
1038 extern void perror(const char *);
1039 extern FILE *popen(const char *, const char *);
1040 extern int printf(const char *, ...);
1041 extern int putc_unlocked(int, FILE *);
1042 extern int putchar(int);
1043 extern int putchar_unlocked(int);
1044 extern int putw(int, FILE *);
1045 extern int remove(const char *);

```

```

1046     extern void rewind(FILE *);
1047     extern int scanf(const char *, ...);
1048     extern void setbuf(FILE *, char *);
1049     extern int sprintf(char *, const char *, ...);
1050     extern int sscanf(const char *, const char *, ...);
1051     extern FILE *stderr(void);
1052     extern FILE *stdin(void);
1053     extern FILE *stdout(void);
1054     extern char *tempnam(const char *, const char *);
1055     extern FILE *tmpfile64(void);
1056     extern FILE *tmpfile(void);
1057     extern char *tmpnam(char *);
1058     extern int vfprintf(FILE *, const char *, va_list);
1059     extern int vprintf(const char *, va_list);
1060     extern int feof(FILE *);
1061     extern int ferror(FILE *);
1062     extern int fflush(FILE *);
1063     extern int fgetc(FILE *);
1064     extern int fgetpos(FILE *, fpos_t *);
1065     extern char *fgets(char *, int, FILE *);
1066     extern int fputs(const char *, FILE *);
1067     extern size_t fread(void *, size_t, size_t, FILE *);
1068     extern int fsetpos(FILE *, const fpos_t *);
1069     extern long int ftell(FILE *);
1070     extern size_t fwrite(const void *, size_t, size_t, FILE *);
1071     extern int getc(FILE *);
1072     extern int putc(int, FILE *);
1073     extern int puts(const char *);
1074     extern int setvbuf(FILE *, char *, int, size_t);
1075     extern int snprintf(char *, size_t, const char *, ...);
1076     extern int ungetc(int, FILE *);
1077     extern int vsnprintf(char *, size_t, const char *, va_list);
1078     extern int vsprintf(char *, const char *, va_list);
1079     extern void flockfile(FILE *);
1080     extern int asprintf(char **, const char *, ...);
1081     extern int fgetpos64(FILE *, fpos64_t *);
1082     extern FILE *fopen64(const char *, const char *);
1083     extern int fsetpos64(FILE *, const fpos64_t *);
1084     extern int ftrylockfile(FILE *);
1085     extern void funlockfile(FILE *);
1086     extern int getc_unlocked(FILE *);
1087     extern void setbuffer(FILE *, char *, size_t);
1088     extern int vasprintf(char **, const char *, va_list);
1089     extern int vdprintf(int, const char *, va_list);
1090     extern int vfscanf(FILE *, const char *, va_list);
1091     extern int vscanf(const char *, va_list);
1092     extern int vsscanf(const char *, const char *, va_list);
1093     extern size_t __fpending(FILE *);

```

11.3.46 stdlib.h

```

1094     extern double __strtod_internal(const char *, char **, int);
1095     extern float __strtof_internal(const char *, char **, int);
1096     extern long int __strtoul_internal(const char *, char **, int, int);
1097     extern long double __strtold_internal(const char *, char **, int);
1098     extern long long int __strtoll_internal(const char *, char **, int, int);
1099     extern unsigned long int __strtoul_internal(const char *, char **, int,
1100                                               int);
1101     extern unsigned long long int __strtoull_internal(const char *, char **,
1102                                                     int, int);
1103     extern long int a64l(const char *);
1104     extern void abort(void);
1105     extern int abs(int);

```

```

1107 extern double atof(const char *);
1108 extern int atoi(char *);
1109 extern long int atol(char *);
1110 extern long long int atoll(const char *);
1111 extern void *bsearch(const void *, const void *, size_t, size_t,
1112     __compar_fn_t);
1113 extern div_t div(int, int);
1114 extern double drand48(void);
1115 extern char *ecvt(double, int, int *, int *);
1116 extern double erand48(unsigned short);
1117 extern void exit(int);
1118 extern char *fcvt(double, int, int *, int *);
1119 extern char *gcvt(double, int, char *);
1120 extern char *getenv(const char *);
1121 extern int getsubopt(char **, char *const *, char **);
1122 extern int grantpt(int);
1123 extern long int jrand48(unsigned short);
1124 extern char *l64a(long int);
1125 extern long int labs(long int);
1126 extern void lcong48(unsigned short);
1127 extern ldiv_t ldiv(long int, long int);
1128 extern long long int llabs(long long int);
1129 extern lldiv_t lldiv(long long int, long long int);
1130 extern long int lrand48(void);
1131 extern int mblen(const char *, size_t);
1132 extern size_t mbstowcs(wchar_t *, const char *, size_t);
1133 extern int mbtowc(wchar_t *, const char *, size_t);
1134 extern char *mktemp(char *);
1135 extern long int mrand48(void);
1136 extern long int nrand48(unsigned short);
1137 extern char *ptsname(int);
1138 extern int putenv(char *);
1139 extern void qsort(void *, size_t, size_t, __compar_fn_t);
1140 extern int rand(void);
1141 extern int rand_r(unsigned int *);
1142 extern unsigned short *seed48(unsigned short);
1143 extern void srand48(long int);
1144 extern int unlockpt(int);
1145 extern size_t wcstombs(char *, const wchar_t *, size_t);
1146 extern int wctomb(char *, wchar_t);
1147 extern int system(const char *);
1148 extern void *calloc(size_t, size_t);
1149 extern void free(void *);
1150 extern char *initstate(unsigned int, char *, size_t);
1151 extern void *malloc(size_t);
1152 extern long int random(void);
1153 extern void *realloc(void *, size_t);
1154 extern char *setstate(char *);
1155 extern void srand(unsigned int);
1156 extern void srand48(unsigned int);
1157 extern double strtod(char *, char **);
1158 extern float strtof(const char *, char **);
1159 extern long int strtol(char *, char **, int);
1160 extern long double strtold(const char *, char **);
1161 extern long long int strtoll(const char *, char **, int);
1162 extern long long int strtouq(const char *, char **, int);
1163 extern unsigned long int strtoul(const char *, char **, int);
1164 extern unsigned long long int strtoull(const char *, char **, int);
1165 extern unsigned long long int strtouq(const char *, char **, int);
1166 extern void _Exit(int);
1167 extern size_t __ctype_get_mb_cur_max(void);
1168 extern char **environ(void);
1169 extern char *realpath(const char *, char *);
1170 extern int setenv(const char *, const char *, int);

```

```

1171     extern int unsetenv(const char *);
1172     extern int getloadavg(double, int);
1173     extern int mkstemp64(char *);
1174     extern int posix_memalign(void **, size_t, size_t);
1175     extern int posix_openpt(int);

```

11.3.47 string.h

```

1176     extern void *__memcpy(void *, const void *, size_t);
1177     extern char *__strcpy(char *, const char *);
1178     extern char *__strtok_r(char *, const char *, char **);
1180     extern void bcopy(void *, void *, size_t);
1181     extern void *memchr(void *, int, size_t);
1182     extern int memcmp(void *, void *, size_t);
1183     extern void *memcpy(void *, void *, size_t);
1184     extern void *memmem(const void *, size_t, const void *, size_t);
1185     extern void *memmove(void *, const void *, size_t);
1186     extern void *memset(void *, int, size_t);
1187     extern char *strcat(char *, const char *);
1188     extern char *strchr(char *, int);
1189     extern int strcmp(char *, char *);
1190     extern int strcoll(const char *, const char *);
1191     extern char *strcpy(char *, char *);
1192     extern size_t strcspn(const char *, const char *);
1193     extern char *strerror(int);
1194     extern size_t strlen(char *);
1195     extern char *strncat(char *, char *, size_t);
1196     extern int strncmp(char *, char *, size_t);
1197     extern char *strncpy(char *, char *, size_t);
1198     extern char *strpbrk(const char *, const char *);
1199     extern char *strrchr(char *, int);
1200     extern char *strsignal(int);
1201     extern size_t strspn(const char *, const char *);
1202     extern char *strstr(char *, char *);
1203     extern char *strtok(char *, const char *);
1204     extern size_t strxfrm(char *, const char *, size_t);
1205     extern int bcmp(void *, void *, size_t);
1206     extern void bzero(void *, size_t);
1207     extern int ffs(int);
1208     extern char *index(char *, int);
1209     extern void *memccpy(void *, const void *, int, size_t);
1210     extern char *rindex(char *, int);
1211     extern int strcasecmp(char *, char *);
1212     extern char *strdup(char *);
1213     extern int strncasecmp(char *, char *, size_t);
1214     extern char *strndup(const char *, size_t);
1215     extern size_t strnlen(const char *, size_t);
1216     extern char *strsep(char **, const char *);
1217     extern char *strerror_r(int, char *, size_t);
1218     extern char *strtok_r(char *, const char *, char **);
1219     extern char *strcasestr(const char *, const char *);
1220     extern char *stpcpy(char *, const char *);
1221     extern char *stpncpy(char *, const char *, size_t);
1222     extern void *memrchr(const void *, int, size_t);

```

11.3.48 sys/file.h

```

1223     extern int flock(int, int);
1224

```


11.3.49 sys/ioctl.h

```

1225
1226 #define TIOCGWINSZ      0x5413
1227 #define FIONREAD       0x541B
1228 #define TIOCNOTTY     0x5422
1229
1230 extern int ioctl(int, unsigned long int, ...);

```

11.3.50 sys/ipc.h

```

1231
1232 struct ipc_perm {
1233     key_t __key;
1234     uid_t uid;
1235     gid_t gid;
1236     uid_t cuid;
1237     uid_t cgid;
1238     mode_t mode;
1239     unsigned short __seq;
1240     unsigned short __pad1;
1241     unsigned long int __unused1;
1242     unsigned long int __unused2;
1243 };
1244
1245 extern key_t ftok(char *, int);

```

11.3.51 sys/mman.h

```

1246
1247 #define MCL_CURRENT     1
1248 #define MCL_FUTURE     2
1249
1250 extern int msync(void *, size_t, int);
1251 extern int mlock(const void *, size_t);
1252 extern int mlockall(int);
1253 extern void *mmap(void *, size_t, int, int, int, off_t);
1254 extern int mprotect(void *, size_t, int);
1255 extern int munlock(const void *, size_t);
1256 extern int munlockall(void);
1257 extern int munmap(void *, size_t);
1258 extern void *mmap64(void *, size_t, int, int, int, off64_t);
1259 extern int shm_open(const char *, int, mode_t);
1260 extern int shm_unlink(const char *);

```

11.3.52 sys/msg.h

```

1261
1262 struct msgid_ds {
1263     struct ipc_perm msg_perm;
1264     time_t msg_stime;
1265     time_t msg_rtime;
1266     time_t msg_ctime;
1267     unsigned long int __msg_cbytes;
1268     unsigned long int msg_qnum;
1269     unsigned long int msg_qbytes;
1270     pid_t msg_lspid;
1271     pid_t msg_lrpid;
1272     unsigned long int __unused1;
1273     unsigned long int __unused2;
1274 };
1275 extern int msgctl(int, int, struct msgid_ds *);
1276 extern int msgget(key_t, int);

```

```

1277 extern int msgrcv(int, void *, size_t, long int, int);
1278 extern int msgsnd(int, const void *, size_t, int);

```

11.3.53 sys/param.h

```

1279
1280 /*
1281  * This header is architecture neutral
1282  * Please refer to the generic specification for details
1283  */

```

11.3.54 sys/poll.h

```

1284
1285 /*
1286  * This header is architecture neutral
1287  * Please refer to the generic specification for details
1288  */

```

11.3.55 sys/resource.h

```

1289
1290 extern int getpriority(__priority_which_t, id_t);
1291 extern int getrlimit64(id_t, struct rlimit64 *);
1292 extern int setpriority(__priority_which_t, id_t, int);
1293 extern int setrlimit(__rlimit_resource_t, const struct rlimit *);
1294 extern int setrlimit64(__rlimit_resource_t, const struct rlimit64 *);
1295 extern int getrlimit(__rlimit_resource_t, struct rlimit *);
1296 extern int getrusage(int, struct rusage *);

```

11.3.56 sys/sem.h

```

1297
1298 struct semid_ds {
1299     struct ipc_perm sem_perm;
1300     time_t sem_otime;
1301     time_t sem_ctime;
1302     unsigned long int sem_nsems;
1303     unsigned long int __unused1;
1304     unsigned long int __unused2;
1305 };
1306 extern int semctl(int, int, int, ...);
1307 extern int semget(key_t, int, int);
1308 extern int semop(int, struct sembuf *, size_t);

```

11.3.57 sys/shm.h

```

1309
1310 #define SHMLBA (1024*1024)
1311
1312 struct shmid_ds {
1313     struct ipc_perm shm_perm;
1314     size_t shm_segsz;
1315     time_t shm_atime;
1316     time_t shm_dtime;
1317     time_t shm_ctime;
1318     pid_t shm_cpid;
1319     pid_t shm_lpid;
1320     unsigned long int shm_nattch;
1321     unsigned long int __unused1;
1322     unsigned long int __unused2;
1323 };
1324 extern int __getpagesize(void);

```

```

1325 extern void *shmat(int, const void *, int);
1326 extern int shmctl(int, int, struct shmid_ds *);
1327 extern int shmdt(const void *);
1328 extern int shmget(key_t, size_t, int);

```

11.3.58 sys/socket.h

```

1329
1330 typedef uint64_t __ss_aligntype;
1331
1332 #define SO_RCVLOWAT      18
1333 #define SO_SNDLOWAT      19
1334 #define SO_RCVTIMEO      20
1335 #define SO_SNDTIMEO      21
1336
1337 extern int bind(int, const struct sockaddr *, socklen_t);
1338 extern int getnameinfo(const struct sockaddr *, socklen_t, char *,
1339                        socklen_t, char *, socklen_t, unsigned int);
1340 extern int getsockname(int, struct sockaddr *, socklen_t *);
1341 extern int listen(int, int);
1342 extern int setsockopt(int, int, int, const void *, socklen_t);
1343 extern int accept(int, struct sockaddr *, socklen_t *);
1344 extern int connect(int, const struct sockaddr *, socklen_t);
1345 extern ssize_t recv(int, void *, size_t, int);
1346 extern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *,
1347                        socklen_t *);
1348 extern ssize_t recvmsg(int, struct msghdr *, int);
1349 extern ssize_t send(int, const void *, size_t, int);
1350 extern ssize_t sendmsg(int, const struct msghdr *, int);
1351 extern ssize_t sendto(int, const void *, size_t, int,
1352                      const struct sockaddr *, socklen_t);
1353 extern int getpeername(int, struct sockaddr *, socklen_t *);
1354 extern int getsockopt(int, int, int, void *, socklen_t *);
1355 extern int shutdown(int, int);
1356 extern int socket(int, int, int);
1357 extern int socketpair(int, int, int, int);
1358 extern int socketatmark(int);

```

11.3.59 sys/stat.h

```

1359
1360 #define _STAT_VER      1
1361
1362 struct stat {
1363     dev_t st_dev;
1364     ino_t st_ino;
1365     nlink_t st_nlink;
1366     mode_t st_mode;
1367     uid_t st_uid;
1368     gid_t st_gid;
1369     unsigned int pad0;
1370     dev_t st_rdev;
1371     off_t st_size;
1372     struct timespec st_atim;
1373     struct timespec st_mtim;
1374     struct timespec st_ctim;
1375     blksize_t st_blksize;
1376     blkcnt_t st_blocks;
1377     unsigned long int __unused[3];
1378 };
1379 struct stat64 {
1380     dev_t st_dev;
1381     ino64_t st_ino;
1382     nlink_t st_nlink;

```

```

1383         mode_t st_mode;
1384         uid_t st_uid;
1385         gid_t st_gid;
1386         unsigned int pad0;
1387         dev_t st_rdev;
1388         off_t st_size;
1389         struct timespec st_atim;
1390         struct timespec st_mtim;
1391         struct timespec st_ctim;
1392         blksize_t st_blksize;
1393         blkcnt64_t st_blocks;
1394         unsigned long int __unused[3];
1395     };
1396
1397     extern int __fxstat(int, int, struct stat *);
1398     extern int __fxstat64(int, int, struct stat64 *);
1399     extern int __lxstat(int, char *, struct stat *);
1400     extern int __lxstat64(int, const char *, struct stat64 *);
1401     extern int __xmknod(int, const char *, mode_t, dev_t *);
1402     extern int __xstat(int, const char *, struct stat *);
1403     extern int __xstat64(int, const char *, struct stat64 *);
1404     extern int mkfifo(const char *, mode_t);
1405     extern int chmod(const char *, mode_t);
1406     extern int fchmod(int, mode_t);
1407     extern mode_t umask(mode_t);

```

11.3.60 sys/statvfs.h

```

1408
1409     struct statvfs {
1410         unsigned long int f_bsize;
1411         unsigned long int f_frsize;
1412         fsblkcnt64_t f_blocks;
1413         fsblkcnt64_t f_bfree;
1414         fsblkcnt64_t f_bavail;
1415         fsfilcnt64_t f_files;
1416         fsfilcnt64_t f_ffree;
1417         fsfilcnt64_t f_favail;
1418         unsigned long int f_fsid;
1419         unsigned long int f_flag;
1420         unsigned long int f_namemax;
1421         unsigned int __f_spare[6];
1422     };
1423     struct statvfs64 {
1424         unsigned long int f_bsize;
1425         unsigned long int f_frsize;
1426         fsblkcnt64_t f_blocks;
1427         fsblkcnt64_t f_bfree;
1428         fsblkcnt64_t f_bavail;
1429         fsfilcnt64_t f_files;
1430         fsfilcnt64_t f_ffree;
1431         fsfilcnt64_t f_favail;
1432         unsigned long int f_fsid;
1433         unsigned long int f_flag;
1434         unsigned long int f_namemax;
1435         unsigned int __f_spare[6];
1436     };
1437     extern int fstatvfs(int, struct statvfs *);
1438     extern int fstatvfs64(int, struct statvfs64 *);
1439     extern int statvfs(const char *, struct statvfs *);
1440     extern int statvfs64(const char *, struct statvfs64 *);

```

11.3.61 sys/time.h

```

1441
1442 extern int getitimer(__itimer_which_t, struct itimerval *);
1443 extern int setitimer(__itimer_which_t, const struct itimerval *,
1444                     struct itimerval *);
1445 extern int adjtime(const struct timeval *, struct timeval *);
1446 extern int gettimeofday(struct timeval *, struct timezone *);
1447 extern int utimes(const char *, const struct timeval *);

```

11.3.62 sys/timeb.h

```

1448
1449 extern int ftime(struct timeb *);

```

11.3.63 sys/times.h

```

1450
1451 extern clock_t times(struct tms *);

```

11.3.64 sys/types.h

```

1452
1453 typedef long int int64_t;
1454
1455 typedef int64_t ssize_t;
1456
1457 #define __FDSET_LONGS 16

```

11.3.65 sys/uio.h

```

1458
1459 extern ssize_t readv(int, const struct iovec *, int);
1460 extern ssize_t writev(int, const struct iovec *, int);

```

11.3.66 sys/un.h

```

1461
1462 /*
1463  * This header is architecture neutral
1464  * Please refer to the generic specification for details
1465  */

```

11.3.67 sys/utsname.h

```

1466
1467 extern int uname(struct utsname *);

```

11.3.68 sys/wait.h

```

1468
1469 extern pid_t wait(int *);
1470 extern pid_t waitpid(pid_t, int *, int);
1471 extern pid_t wait4(pid_t, int *, int, struct rusage *);

```

11.3.69 syslog.h

```

1472
1473 extern void closelog(void);
1474 extern void openlog(const char *, int, int);
1475 extern int setlogmask(int);

```

```

1476     extern void syslog(int, const char *, ...);
1477     extern void vsyslog(int, const char *, va_list);

```

11.3.70 termios.h

```

1478
1479     #define OLCUC      0000002
1480     #define ONLCR     0000004
1481     #define XCASE     0000004
1482     #define NLDLY     0000400
1483     #define CR1       0001000
1484     #define IUCLC     0001000
1485     #define CR2       0002000
1486     #define CR3       0003000
1487     #define CRDLY     0003000
1488     #define TAB1      0004000
1489     #define TAB2      0010000
1490     #define TAB3      0014000
1491     #define TABDLY    0014000
1492     #define BS1       0020000
1493     #define BSDLY     0020000
1494     #define VT1       0040000
1495     #define VTDLY     0040000
1496     #define FF1       0100000
1497     #define FFDLY     0100000
1498
1499     #define VSUSP      10
1500     #define VEOL       11
1501     #define VREPRINT   12
1502     #define VDISCARD   13
1503     #define VWERASE    14
1504     #define VEOL2     16
1505     #define VMIN       6
1506     #define VSWTC      7
1507     #define VSTART     8
1508     #define VSTOP      9
1509
1510     #define IXON       0002000
1511     #define IXOFF      0010000
1512
1513     #define CS6        0000020
1514     #define CS7        0000040
1515     #define CS8        0000060
1516     #define CSIZE      0000060
1517     #define CSTOPB     0000100
1518     #define CREAD      0000200
1519     #define PARENB     0000400
1520     #define PARODD     0001000
1521     #define HUPCL      0002000
1522     #define CLOCAL     0004000
1523     #define VTIME      5
1524
1525     #define ISIG       0000001
1526     #define ICANON     0000002
1527     #define ECHOE      0000020
1528     #define ECHOK      0000040
1529     #define ECHONL     0000100
1530     #define NOFLSH     0000200
1531     #define TOSTOP     0000400
1532     #define ECHOCTL    0001000
1533     #define ECHOPRT    0002000
1534     #define ECHOKE     0004000
1535     #define FLUSHO     0010000
1536     #define PENDIN     0040000

```

```

1537 #define IEXTEN 0100000
1538
1539 extern speed_t cfgetispeed(const struct termios *);
1540 extern speed_t cfgetospeed(const struct termios *);
1541 extern void cfmakeraw(struct termios *);
1542 extern int cfsetispeed(struct termios *, speed_t);
1543 extern int cfsetospeed(struct termios *, speed_t);
1544 extern int cfsetspeed(struct termios *, speed_t);
1545 extern int tcflow(int, int);
1546 extern int tcflush(int, int);
1547 extern pid_t tcgetsid(int);
1548 extern int tcsendbreak(int, int);
1549 extern int tcsetattr(int, int, const struct termios *);
1550 extern int tcdrain(int);
1551 extern int tcgetattr(int, struct termios *);

```

11.3.71 time.h

```

1552
1553 extern int __daylight(void);
1554 extern long int __timezone(void);
1555 extern char *__tzname(void);
1556 extern char *asctime(const struct tm *);
1557 extern clock_t clock(void);
1558 extern char *ctime(const time_t *);
1559 extern char *ctime_r(const time_t *, char *);
1560 extern double difftime(time_t, time_t);
1561 extern struct tm *getdate(const char *);
1562 extern int getdate_err(void);
1563 extern struct tm *gmtime(const time_t *);
1564 extern struct tm *localtime(const time_t *);
1565 extern time_t mktime(struct tm *);
1566 extern int stime(const time_t *);
1567 extern size_t strftime(char *, size_t, const char *, const struct tm *);
1568 extern char *strptime(const char *, const char *, struct tm *);
1569 extern time_t time(time_t *);
1570 extern int nanosleep(const struct timespec *, struct timespec *);
1571 extern int daylight(void);
1572 extern long int timezone(void);
1573 extern char *tzname(void);
1574 extern void tzset(void);
1575 extern char *asctime_r(const struct tm *, char *);
1576 extern struct tm *gmtime_r(const time_t *, struct tm *);
1577 extern struct tm *localtime_r(const time_t *, struct tm *);
1578 extern int clock_getcpu(clockid_t, clockid_t *);
1579 extern int clock_getres(clockid_t, struct timespec *);
1580 extern int clock_gettime(clockid_t, struct timespec *);
1581 extern int clock_nanosleep(clockid_t, int, const struct timespec *,
1582                             struct timespec *);
1583 extern int clock_settime(clockid_t, const struct timespec *);
1584 extern int timer_create(clockid_t, struct sigevent *, timer_t *);
1585 extern int timer_delete(timer_t);
1586 extern int timer_getoverrun(timer_t);
1587 extern int timer_gettime(timer_t, struct itimerspec *);
1588 extern int timer_settime(timer_t, int, const struct itimerspec *,
1589                             struct itimerspec *);

```

11.3.72 ucontext.h

```

1590
1591 #define _SC_GRO_OFFSET \
1592     (((char *) &((struct sigcontext *) 0)->sc_gr[0]) - (char *) 0)
1593
1594 typedef struct sigcontext mcontext_t;

```

```

1595
1596     typedef struct ucontext {
1597         union {
1598             mcontext_t _mc;
1599             struct {
1600                 unsigned long int _pad[_SC_GR0_OFFSET / 8];
1601                 struct ucontext *_link;
1602             } _uc;
1603         } _u;
1604     } ucontext_t;
1605     extern int getcontext(ucontext_t *);
1606     extern int makecontext(ucontext_t *, void (*func) (void)
1607                          , int, ...);
1608     extern int setcontext(const struct ucontext *);
1609     extern int swapcontext(ucontext_t *, const struct ucontext *);

```

11.3.73 ulimit.h

```

1610
1611     extern long int ulimit(int, ...);

```

11.3.74 unistd.h

```

1612
1613     typedef long int intptr_t;
1614
1615     extern char **__environ(void);
1616     extern pid_t __getpgid(pid_t);
1617     extern void _exit(int);
1618     extern int acct(const char *);
1619     extern unsigned int alarm(unsigned int);
1620     extern int chown(const char *, uid_t, gid_t);
1621     extern int chroot(const char *);
1622     extern size_t confstr(int, char *, size_t);
1623     extern int creat(const char *, mode_t);
1624     extern int creat64(const char *, mode_t);
1625     extern char *ctermid(char *);
1626     extern char *cuserid(char *);
1627     extern int daemon(int, int);
1628     extern int execl(const char *, const char *, ...);
1629     extern int execlp(const char *, const char *, ...);
1630     extern int execlp(const char *, const char *, ...);
1631     extern int execv(const char *, char *const);
1632     extern int execvp(const char *, char *const);
1633     extern int fdatsync(int);
1634     extern int ftruncate64(int, off64_t);
1635     extern long int gethostid(void);
1636     extern char *getlogin(void);
1637     extern int getlogin_r(char *, size_t);
1638     extern int getopt(int, char *const, const char *);
1639     extern pid_t getpgrp(void);
1640     extern pid_t getsid(pid_t);
1641     extern char *getwd(char *);
1642     extern int lockf(int, int, off_t);
1643     extern int mkstemp(char *);
1644     extern int nice(int);
1645     extern char *optarg(void);
1646     extern int opterr(void);
1647     extern int optind(void);
1648     extern int optopt(void);
1649     extern int rename(const char *, const char *);
1650     extern int setegid(gid_t);
1651     extern int seteuid(uid_t);
1652     extern int sethostname(const char *, size_t);

```



```

1653     extern int setpgrp(void);
1654     extern void swab(const void *, void *, ssize_t);
1655     extern void sync(void);
1656     extern pid_t tcgetpgrp(int);
1657     extern int tcsetpgrp(int, pid_t);
1658     extern int truncate(const char *, off_t);
1659     extern int truncate64(const char *, off64_t);
1660     extern char *ttyname(int);
1661     extern unsigned int ualarm(useconds_t, useconds_t);
1662     extern int usleep(useconds_t);
1663     extern int close(int);
1664     extern int fsync(int);
1665     extern off_t lseek(int, off_t, int);
1666     extern int open(const char *, int, ...);
1667     extern int pause(void);
1668     extern ssize_t read(int, void *, size_t);
1669     extern ssize_t write(int, const void *, size_t);
1670     extern char *crypt(char *, char *);
1671     extern void encrypt(char *, int);
1672     extern void setkey(const char *);
1673     extern int access(const char *, int);
1674     extern int brk(void *);
1675     extern int chdir(const char *);
1676     extern int dup(int);
1677     extern int dup2(int, int);
1678     extern int execve(const char *, char *const, char *const);
1679     extern int fchdir(int);
1680     extern int fchown(int, uid_t, gid_t);
1681     extern pid_t fork(void);
1682     extern gid_t getegid(void);
1683     extern uid_t geteuid(void);
1684     extern gid_t getgid(void);
1685     extern int getgroups(int, gid_t);
1686     extern int gethostname(char *, size_t);
1687     extern pid_t getpgid(pid_t);
1688     extern pid_t getpid(void);
1689     extern uid_t getuid(void);
1690     extern int lchown(const char *, uid_t, gid_t);
1691     extern int link(const char *, const char *);
1692     extern int mkdir(const char *, mode_t);
1693     extern long int pathconf(const char *, int);
1694     extern int pipe(int);
1695     extern int readlink(const char *, char *, size_t);
1696     extern int rmdir(const char *);
1697     extern void *sbrk(ptrdiff_t);
1698     extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
1699     extern int setgid(gid_t);
1700     extern int setpgid(pid_t, pid_t);
1701     extern int setregid(gid_t, gid_t);
1702     extern int setreuid(uid_t, uid_t);
1703     extern pid_t setsid(void);
1704     extern int setuid(uid_t);
1705     extern unsigned int sleep(unsigned int);
1706     extern int symlink(const char *, const char *);
1707     extern long int sysconf(int);
1708     extern int unlink(const char *);
1709     extern pid_t vfork(void);
1710     extern ssize_t pread(int, void *, size_t, off_t);
1711     extern ssize_t pwrite(int, const void *, size_t, off_t);
1712     extern char **_environ(void);
1713     extern long int fpathconf(int, int);
1714     extern int ftruncate(int, off_t);
1715     extern char *getcwd(char *, size_t);
1716     extern int getpagesize(void);

```

```

1717 extern pid_t getppid(void);
1718 extern int isatty(int);
1719 extern loff_t lseek64(int, loff_t, int);
1720 extern int open64(const char *, int, ...);
1721 extern ssize_t pread64(int, void *, size_t, off64_t);
1722 extern ssize_t pwrite64(int, const void *, size_t, off64_t);
1723 extern int ttyname_r(int, char *, size_t);

```

11.3.75 utime.h

```

1724 extern int utime(const char *, const struct utimbuf *);
1725

```

11.3.76 utmp.h

```

1726
1727 struct lastlog {
1728     time_t ll_time;
1729     char ll_line[UT_LINESIZE];
1730     char ll_host[UT_HOSTSIZE];
1731 };
1732
1733 struct utmp {
1734     short ut_type;
1735     pid_t ut_pid;
1736     char ut_line[UT_LINESIZE];
1737     char ut_id[4];
1738     char ut_user[UT_NAMESIZE];
1739     char ut_host[UT_HOSTSIZE];
1740     struct exit_status ut_exit;
1741     long int ut_session;
1742     struct timeval ut_tv;
1743     int32_t ut_addr_v6[4];
1744     char __unused[20];
1745 };
1746
1747 extern void endutent(void);
1748 extern struct utmp *getutent(void);
1749 extern void setutent(void);
1750 extern int getutent_r(struct utmp *, struct utmp **);
1751 extern int utmpname(const char *);
1752 extern int login_tty(int);
1753 extern void login(const struct utmp *);
1754 extern int logout(const char *);
1755 extern void logwtmp(const char *, const char *, const char *);

```

11.3.77 utmpx.h

```

1756
1757 struct utmpx {
1758     short ut_type;
1759     pid_t ut_pid;
1760     char ut_line[UT_LINESIZE];
1761     char ut_id[4];
1762     char ut_user[UT_NAMESIZE];
1763     char ut_host[UT_HOSTSIZE];
1764     struct exit_status ut_exit;
1765     long int ut_session;
1766     struct timeval ut_tv;
1767     int32_t ut_addr_v6[4];
1768     char __unused[20];
1769 };
1770

```

```

1771 extern void endutxent(void);
1772 extern struct utmpx *getutxent(void);
1773 extern struct utmpx *getutxid(const struct utmpx *);
1774 extern struct utmpx *getutxline(const struct utmpx *);
1775 extern struct utmpx *pututxline(const struct utmpx *);
1776 extern void setutxent(void);

```

11.3.78 wchar.h

```

1777
1778 extern double __wcstod_internal(const wchar_t *, wchar_t **, int);
1779 extern float __wcstof_internal(const wchar_t *, wchar_t **, int);
1780 extern long int __wcstol_internal(const wchar_t *, wchar_t **, int,
1781 int);
1782 extern long double __wcstold_internal(const wchar_t *, wchar_t **, int);
1783 extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t *
1784 *,
1785 int, int);
1786 extern wchar_t *wscat(wchar_t *, const wchar_t *);
1787 extern wchar_t *wcschr(const wchar_t *, wchar_t);
1788 extern int wcscmp(const wchar_t *, const wchar_t *);
1789 extern int wscoll(const wchar_t *, const wchar_t *);
1790 extern wchar_t *wcscpy(wchar_t *, const wchar_t *);
1791 extern size_t wcsncpy(const wchar_t *, const wchar_t *);
1792 extern wchar_t *wcsdup(const wchar_t *);
1793 extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
1794 extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
1795 extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
1796 extern wchar_t *wcpbrk(const wchar_t *, const wchar_t *);
1797 extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
1798 extern size_t wcsspn(const wchar_t *, const wchar_t *);
1799 extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
1800 extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t **);
1801 extern int wcswidth(const wchar_t *, size_t);
1802 extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
1803 extern int wctob(wint_t);
1804 extern int wcwidth(wchar_t);
1805 extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
1806 extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
1807 extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
1808 extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
1809 extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
1810 extern size_t mbrlen(const char *, size_t, mbstate_t *);
1811 extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
1812 extern int mbsinit(const mbstate_t *);
1813 extern size_t mbsnrtowcs(wchar_t *, const char **, size_t, size_t,
1814 mbstate_t *);
1815 extern size_t mbsrtowcs(wchar_t *, const char **, size_t, mbstate_t *);
1816 extern wchar_t *wcpncpy(wchar_t *, const wchar_t *);
1817 extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
1818 extern size_t wctomb(char *, wchar_t, mbstate_t *);
1819 extern size_t wcslen(const wchar_t *);
1820 extern size_t wcsnrtombs(char *, const wchar_t **, size_t, size_t,
1821 mbstate_t *);
1822 extern size_t wcsrtombs(char *, const wchar_t **, size_t, mbstate_t *);
1823 extern double wcstod(const wchar_t *, wchar_t **);
1824 extern float wcstof(const wchar_t *, wchar_t **);
1825 extern long int wcstol(const wchar_t *, wchar_t **, int);
1826 extern long double wcstold(const wchar_t *, wchar_t **);
1827 extern long long int wcstoq(const wchar_t *, wchar_t **, int);
1828 extern unsigned long int wcstoul(const wchar_t *, wchar_t **, int);
1829 extern unsigned long long int wcstouq(const wchar_t *, wchar_t **, int);
1830 extern wchar_t *wswcs(const wchar_t *, const wchar_t *);
1831 extern int wscasecmp(const wchar_t *, const wchar_t *);

```

```

1832     extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
1833     extern size_t wcsnlen(const wchar_t *, size_t);
1834     extern long long int wcstoll(const wchar_t *, wchar_t ** , int);
1835     extern unsigned long long int wcstoull(const wchar_t *, wchar_t ** , int);
1836     extern wint_t btowc(int);
1837     extern wint_t fgetwc(FILE *);
1838     extern wint_t fgetwc_unlocked(FILE *);
1839     extern wchar_t *fgetws(wchar_t *, int, FILE *);
1840     extern wint_t fputwc(wchar_t, FILE *);
1841     extern int fputws(const wchar_t *, FILE *);
1842     extern int fwide(FILE *, int);
1843     extern int fwprintf(FILE *, const wchar_t *, ...);
1844     extern int fwscanf(FILE *, const wchar_t *, ...);
1845     extern wint_t getwc(FILE *);
1846     extern wint_t getwchar(void);
1847     extern wint_t putwc(wchar_t, FILE *);
1848     extern wint_t putwchar(wchar_t);
1849     extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
1850     extern int swscanf(const wchar_t *, const wchar_t *, ...);
1851     extern wint_t ungetwc(wint_t, FILE *);
1852     extern int vfwprintf(FILE *, const wchar_t *, va_list);
1853     extern int vfwscanf(FILE *, const wchar_t *, va_list);
1854     extern int vswprintf(wchar_t *, size_t, const wchar_t *, va_list);
1855     extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
1856     extern int vwprintf(const wchar_t *, va_list);
1857     extern int vwscanf(const wchar_t *, va_list);
1858     extern size_t wcsftime(wchar_t *, size_t, const wchar_t *,
1859                           const struct tm *);
1860     extern int wprintf(const wchar_t *, ...);
1861     extern int wscanf(const wchar_t *, ...);

```

11.3.79 wctype.h

```

1862     extern int iswblank(wint_t);
1863     extern wint_t towlower(wint_t);
1864     extern wint_t towupper(wint_t);
1865     extern wctrans_t wctrans(const char *);
1866     extern int iswalnum(wint_t);
1867     extern int iswalpha(wint_t);
1868     extern int iswcntrl(wint_t);
1869     extern int iswctype(wint_t, wctype_t);
1870     extern int iswdigit(wint_t);
1871     extern int iswgraph(wint_t);
1872     extern int iswlower(wint_t);
1873     extern int iswprint(wint_t);
1874     extern int iswpunct(wint_t);
1875     extern int iswspace(wint_t);
1876     extern int iswupper(wint_t);
1877     extern int iswxdigit(wint_t);
1878     extern wctype_t wctype(const char *);
1879     extern wint_t towctrans(wint_t, wctrans_t);

```

11.3.80 wordexp.h

```

1881     extern int wordexp(const char *, wordexp_t *, int);
1882     extern void wordfree(wordexp_t *);
1883

```

11.4 Interfaces for libm

1884 Table 11-24 defines the library name and shared object name for the libm library

1885

Table 11-24 libm Definition

| | |
|----------|-------------|
| Library: | libm |
| SONAME: | libm.so.6.1 |

1886

1887

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

1888

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

1889

11.4.1 Math

1890

11.4.1.1 Interfaces for Math

1891

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

1892

1893

1894

Table 11-25 libm - Math Function Interfaces

| | | | | |
|---|----------------------------------|----------------------------------|-------------------------------------|--------------------------------------|
| __finite(GLIBC_2.2) [1] | ecoshl(GLIBC_2.2) [2] | exp(GLIBC_2.2) [2] | j1(GLIBC_2.2) [1] | powl(GLIBC_2.2) [2] |
| __finitef(GLIBC_2.2) [1] | ecosh(GLIBC_2.2) [2] | exp2(GLIBC_2.2) [2] | jn(GLIBC_2.2) [2] | remainder(GLIBC_2.2) [2] |
| __finitel(GLIBC_2.2) [1] | ceil(GLIBC_2.2) [2] | exp2f(GLIBC_2.2) [2] | jnf(GLIBC_2.2) [1] | remainderf(GLIBC_2.2) [2] |
| __fpclassify(GLIBC_2.2) [3] | ceilf(GLIBC_2.2) [2] | exp2l(GLIBC_2.2) [2] | jnl(GLIBC_2.2) [1] | remainderl(GLIBC_2.2) [2] |
| __fpclassifyf(GLIBC_2.2) [3] | ceilll(GLIBC_2.2) [2] | expf(GLIBC_2.2) [2] | ldexp(GLIBC_2.2) [2] | remquo(GLIBC_2.2) [2] |
| __fpclassifyl(GLIBC_2.2) [1] | eexp(GLIBC_2.2) [2] | expl(GLIBC_2.2) [2] | ldexpf(GLIBC_2.2) [2] | remquof(GLIBC_2.2) [2] |
| __signbit(GLIBC_2.2) [1] | eexpf(GLIBC_2.2) [2] | expm1(GLIBC_2.2) [2] | ldexpl(GLIBC_2.2) [2] | remquol(GLIBC_2.2) [2] |
| __signbitf(GLIBC_2.2) [1] | expl(GLIBC_2.2) [2] | expm1f(GLIBC_2.2) [2] | lgamma(GLIBC_2.2) [2] | rint(GLIBC_2.2) [2] |
| __signbitl(GLIBC_2.2) [1] | imag(GLIBC_2.2) [2] | expm1l(GLIBC_2.2) [2] | lgamma_r(GLIBC_2.2) [1] | rintf(GLIBC_2.2) [2] |
| acos(GLIBC_2.2) [2] | imagf(GLIBC_2.2) [2] | fabs(GLIBC_2.2) [2] | lgammaf(GLIBC_2.2) [2] | rintl(GLIBC_2.2) [2] |
| acosf(GLIBC_2.2) [2] | imagl(GLIBC_2.2) [2] | fabsf(GLIBC_2.2) [2] | lgammaf_r(GLIBC_2.2) [1] | round(GLIBC_2.2) [2] |
| acosh(GLIBC_2.2) [2] | elog(GLIBC_2.2) [2] | fabsl(GLIBC_2.2) [2] | lgammal(GLIBC_2.2) [2] | roundf(GLIBC_2.2) [2] |

| | | | | |
|--------------------------------------|---|---|--|--|
| <code>_{2.2}) [2]</code> | <code>_{2}) [2]</code> | <code>_{2.2}) [2]</code> | <code>BC_{2.2}) [2]</code> | <code>_{2.2}) [2]</code> |
| <code>acoshf(GLIBC_{2.2}) [2]</code> | <code>elog10(GLIBC_{2.2}) [1]</code> | <code>fdim(GLIBC_{2.2}) [2]</code> | <code>lgamma_r(GLIBC_{2.2}) [1]</code> | <code>roundl(GLIBC_{2.2}) [2]</code> |
| <code>acoshl(GLIBC_{2.2}) [2]</code> | <code>elog10f(GLIBC_{2.2}) [1]</code> | <code>fdimf(GLIBC_{2.2}) [2]</code> | <code>llrint(GLIBC_{2.2}) [2]</code> | <code>scalb(GLIBC_{2.2}) [2]</code> |
| <code>acosl(GLIBC_{2.2}) [2]</code> | <code>elog10l(GLIBC_{2.2}) [1]</code> | <code>fdiml(GLIBC_{2.2}) [2]</code> | <code>llrintf(GLIBC_{2.2}) [2]</code> | <code>scalbf(GLIBC_{2.2}) [1]</code> |
| <code>asin(GLIBC_{2.2}) [2]</code> | <code>elogf(GLIBC_{2.2}) [2]</code> | <code>feclearexcept(GLIBC_{2.2}) [2]</code> | <code>llrintl(GLIBC_{2.2}) [2]</code> | <code>scalbl(GLIBC_{2.2}) [1]</code> |
| <code>asinf(GLIBC_{2.2}) [2]</code> | <code>elogl(GLIBC_{2.2}) [2]</code> | <code>fegetenv(GLIBC_{2.2}) [2]</code> | <code>llround(GLIBC_{2.2}) [2]</code> | <code>scalbln(GLIBC_{2.2}) [2]</code> |
| <code>asinh(GLIBC_{2.2}) [2]</code> | <code>conj(GLIBC_{2.2}) [2]</code> | <code>fegetexceptflag(GLIBC_{2.2}) [2]</code> | <code>llroundf(GLIBC_{2.2}) [2]</code> | <code>scalblnf(GLIBC_{2.2}) [2]</code> |
| <code>asinhf(GLIBC_{2.2}) [2]</code> | <code>conjf(GLIBC_{2.2}) [2]</code> | <code>fegetround(GLIBC_{2.2}) [2]</code> | <code>llroundl(GLIBC_{2.2}) [2]</code> | <code>scalblnl(GLIBC_{2.2}) [2]</code> |
| <code>asinhL(GLIBC_{2.2}) [2]</code> | <code>conjl(GLIBC_{2.2}) [2]</code> | <code>fehldexcept(GLIBC_{2.2}) [2]</code> | <code>log(GLIBC_{2.2}) [2]</code> | <code>scalbn(GLIBC_{2.2}) [2]</code> |
| <code>asinl(GLIBC_{2.2}) [2]</code> | <code>copysign(GLIBC_{2.2}) [2]</code> | <code>feraiseexcept(GLIBC_{2.2}) [2]</code> | <code>log10(GLIBC_{2.2}) [2]</code> | <code>scalbnf(GLIBC_{2.2}) [2]</code> |
| <code>atan(GLIBC_{2.2}) [2]</code> | <code>copysignf(GLIBC_{2.2}) [2]</code> | <code>fesetenv(GLIBC_{2.2}) [2]</code> | <code>log10f(GLIBC_{2.2}) [2]</code> | <code>scalbnl(GLIBC_{2.2}) [2]</code> |
| <code>atan2(GLIBC_{2.2}) [2]</code> | <code>copysignl(GLIBC_{2.2}) [2]</code> | <code>fesetexceptflag(GLIBC_{2.2}) [2]</code> | <code>log10l(GLIBC_{2.2}) [2]</code> | <code>significand(GLIBC_{2.2}) [1]</code> |
| <code>atan2f(GLIBC_{2.2}) [2]</code> | <code>cos(GLIBC_{2.2}) [2]</code> | <code>fesetround(GLIBC_{2.2}) [2]</code> | <code>log1p(GLIBC_{2.2}) [2]</code> | <code>significandf(GLIBC_{2.2}) [1]</code> |
| <code>atan2l(GLIBC_{2.2}) [2]</code> | <code>cosf(GLIBC_{2.2}) [2]</code> | <code>fetestexcept(GLIBC_{2.2}) [2]</code> | <code>log1pf(GLIBC_{2.2}) [2]</code> | <code>significandl(GLIBC_{2.2}) [1]</code> |
| <code>atanf(GLIBC_{2.2}) [2]</code> | <code>cosh(GLIBC_{2.2}) [2]</code> | <code>feupdateenv(GLIBC_{2.2}) [2]</code> | <code>log1pl(GLIBC_{2.2}) [2]</code> | <code>sin(GLIBC_{2.2}) [2]</code> |
| <code>atanh(GLIBC_{2.2}) [2]</code> | <code>coshf(GLIBC_{2.2}) [2]</code> | <code>finite(GLIBC_{2.2}) [4]</code> | <code>log2(GLIBC_{2.2}) [2]</code> | <code>sincos(GLIBC_{2.2}) [1]</code> |
| <code>atanhf(GLIBC_{2.2}) [2]</code> | <code>coshl(GLIBC_{2.2}) [2]</code> | <code>finitef(GLIBC_{2.2}) [1]</code> | <code>log2f(GLIBC_{2.2}) [2]</code> | <code>sincosf(GLIBC_{2.2}) [1]</code> |
| <code>atanhl(GLIBC_{2.2}) [2]</code> | <code>cosl(GLIBC_{2.2}) [2]</code> | <code>finitel(GLIBC_{2.2}) [1]</code> | <code>log2l(GLIBC_{2.2}) [2]</code> | <code>sincosl(GLIBC_{2.2}) [1]</code> |
| <code>atanl(GLIBC_{2.2}) [2]</code> | <code>epow(GLIBC_{2.2}) [2]</code> | <code>floor(GLIBC_{2.2}) [2]</code> | <code>logb(GLIBC_{2.2}) [2]</code> | <code>sinf(GLIBC_{2.2}) [2]</code> |

| | | | | |
|------------------------|------------------------|------------------------|---------------------------|-------------------------|
| 2.2) [2] | 2.2) [2] | 2.2) [2] | 2) [2] | 2) [2] |
| cabs(GLIBC_2.2) [2] | epowf(GLIBC_2.2) [2] | floorf(GLIBC_2.2) [2] | logbf(GLIBC_2.2) [2] | sinh(GLIBC_2.2) [2] |
| cabsf(GLIBC_2.2) [2] | epowl(GLIBC_2.2) [2] | floorl(GLIBC_2.2) [2] | logbl(GLIBC_2.2) [2] | sinhf(GLIBC_2.2) [2] |
| cabsl(GLIBC_2.2) [2] | eprojl(GLIBC_2.2) [2] | fmaf(GLIBC_2.2) [2] | logf(GLIBC_2.2) [2] | sinhl(GLIBC_2.2) [2] |
| cacos(GLIBC_2.2) [2] | eprojl(GLIBC_2.2) [2] | fmaf(GLIBC_2.2) [2] | logl(GLIBC_2.2) [2] | sinl(GLIBC_2.2) [2] |
| cacosf(GLIBC_2.2) [2] | eprojl(GLIBC_2.2) [2] | fmal(GLIBC_2.2) [2] | lrint(GLIBC_2.2) [2] | sqrt(GLIBC_2.2) [2] |
| cacosh(GLIBC_2.2) [2] | erealf(GLIBC_2.2) [2] | fmax(GLIBC_2.2) [2] | lrintf(GLIBC_2.2) [2] | sqrtf(GLIBC_2.2) [2] |
| cacoshf(GLIBC_2.2) [2] | erealf(GLIBC_2.2) [2] | fmaxf(GLIBC_2.2) [2] | lrintl(GLIBC_2.2) [2] | sqrtil(GLIBC_2.2) [2] |
| cacoshl(GLIBC_2.2) [2] | ereall(GLIBC_2.2) [2] | fmaxl(GLIBC_2.2) [2] | lround(GLIBC_2.2) [2] | tan(GLIBC_2.2) [2] |
| cacosl(GLIBC_2.2) [2] | esin(GLIBC_2.2) [2] | fmin(GLIBC_2.2) [2] | lroundf(GLIBC_2.2) [2] | tanf(GLIBC_2.2) [2] |
| carg(GLIBC_2.2) [2] | esinf(GLIBC_2.2) [2] | fminf(GLIBC_2.2) [2] | lroundl(GLIBC_2.2) [2] | tanh(GLIBC_2.2) [2] |
| cargf(GLIBC_2.2) [2] | esinh(GLIBC_2.2) [2] | fminl(GLIBC_2.2) [2] | matherr(GLIBC_2.2) [1] | tanhf(GLIBC_2.2) [2] |
| cargl(GLIBC_2.2) [2] | esinhf(GLIBC_2.2) [2] | fmod(GLIBC_2.2) [2] | modf(GLIBC_2.2) [2] | tanhl(GLIBC_2.2) [2] |
| casin(GLIBC_2.2) [2] | esinhl(GLIBC_2.2) [2] | fmodf(GLIBC_2.2) [2] | modff(GLIBC_2.2) [2] | tanl(GLIBC_2.2) [2] |
| casinf(GLIBC_2.2) [2] | esinl(GLIBC_2.2) [2] | fmodl(GLIBC_2.2) [2] | modfl(GLIBC_2.2) [2] | tgamma(GLIBC_2.2) [2] |
| casinh(GLIBC_2.2) [2] | esqrt(GLIBC_2.2) [2] | frexp(GLIBC_2.2) [2] | nan(GLIBC_2.2) [2] | tgammaf(GLIBC_2.2) [2] |
| casinhf(GLIBC_2.2) [2] | esqrtf(GLIBC_2.2) [2] | frexpf(GLIBC_2.2) [2] | nanf(GLIBC_2.2) [2] | tgammalf(GLIBC_2.2) [2] |
| casinhl(GLIBC_2.2) [2] | esqrtil(GLIBC_2.2) [2] | frexpl(GLIBC_2.2) [2] | nanl(GLIBC_2.2) [2] | trunc(GLIBC_2.2) [2] |
| casinl(GLIBC_2.2) [2] | etan(GLIBC_2.2) [2] | gamma(GLIBC_2.2) [4] | nearbyint(GLIBC_2.2) [2] | truncf(GLIBC_2.2) [2] |
| catan(GLIBC_2.2) [2] | etanf(GLIBC_2.2) [2] | gammaf(GLIBC_2.2) [1] | nearbyintf(GLIBC_2.2) [2] | truncl(GLIBC_2.2) [2] |
| catanf(GLIBC_2.2) [2] | etanh(GLIBC_2.2) [2] | gammalf(GLIBC_2.2) [1] | nearbyintl(GLIBC_2.2) [2] | y0(GLIBC_2.2) [2] |

| | | | | |
|-------------------------------------|------------------------------------|------------------------------------|---|---------------------------------|
| <code>catanh(GLIBC_2.2)</code> [2] | <code>etanhf(GLIBC_2.2)</code> [2] | <code>hypot(GLIBC_2.2)</code> [2] | <code>nextafter(GLIBC_2.2)</code> [2] | <code>y0f(GLIBC_2.2)</code> [1] |
| <code>catanhf(GLIBC_2.2)</code> [2] | <code>etanh(GLIBC_2.2)</code> [2] | <code>hypotf(GLIBC_2.2)</code> [2] | <code>nextafterf(GLIBC_2.2)</code> [2] | <code>y0l(GLIBC_2.2)</code> [1] |
| <code>catanh(GLIBC_2.2)</code> [2] | <code>etanh(GLIBC_2.2)</code> [2] | <code>hypotl(GLIBC_2.2)</code> [2] | <code>nextafterl(GLIBC_2.2)</code> [2] | <code>y1(GLIBC_2.2)</code> [2] |
| <code>catanl(GLIBC_2.2)</code> [2] | <code>dremf(GLIBC_2.2)</code> [1] | <code>ilogb(GLIBC_2.2)</code> [2] | <code>nexttoward(GLIBC_2.2)</code> [2] | <code>y1f(GLIBC_2.2)</code> [1] |
| <code>cbtrf(GLIBC_2.2)</code> [2] | <code>dremf(GLIBC_2.2)</code> [1] | <code>ilogbf(GLIBC_2.2)</code> [2] | <code>nexttowardf(GLIBC_2.2)</code> [2] | <code>y1l(GLIBC_2.2)</code> [1] |
| <code>cbtrf(GLIBC_2.2)</code> [2] | <code>erf(GLIBC_2.2)</code> [2] | <code>ilogbl(GLIBC_2.2)</code> [2] | <code>nexttowardl(GLIBC_2.2)</code> [2] | <code>yn(GLIBC_2.2)</code> [2] |
| <code>cbtrl(GLIBC_2.2)</code> [2] | <code>erfc(GLIBC_2.2)</code> [2] | <code>j0(GLIBC_2.2)</code> [2] | <code>pow(GLIBC_2.2)</code> [2] | <code>ynf(GLIBC_2.2)</code> [1] |
| <code>ceos(GLIBC_2.2)</code> [2] | <code>erfcf(GLIBC_2.2)</code> [2] | <code>j0f(GLIBC_2.2)</code> [1] | <code>pow10(GLIBC_2.2)</code> [1] | <code>ynl(GLIBC_2.2)</code> [1] |
| <code>ceosf(GLIBC_2.2)</code> [2] | <code>erfcf(GLIBC_2.2)</code> [2] | <code>j0l(GLIBC_2.2)</code> [1] | <code>pow10f(GLIBC_2.2)</code> [1] | |
| <code>ceosh(GLIBC_2.2)</code> [2] | <code>erff(GLIBC_2.2)</code> [2] | <code>j1(GLIBC_2.2)</code> [2] | <code>pow10l(GLIBC_2.2)</code> [1] | |
| <code>ceoshf(GLIBC_2.2)</code> [2] | <code>erfl(GLIBC_2.2)</code> [2] | <code>j1f(GLIBC_2.2)</code> [1] | <code>powf(GLIBC_2.2)</code> [2] | |

1895

1896

1897

Referenced Specification(s)

[1]

| | | | |
|--|---|---|--|
| <code>__finite(GLIBC_2)</code> [ISO C99] | <code>__finitef(GLIBC_2.2)</code> [ISO C99] | <code>__finitel(GLIBC_2.2)</code> [ISO C99] | <code>__fpclassify(GLIBC_2.2)</code> [LSB] |
| <code>__fpclassifyf(GLIBC_2.2)</code> [LSB] | <code>__fpclassifyl(GLIBC_2.2)</code> [ISO C99] | <code>__signbit(GLIBC_2.2)</code> [ISO C99] | <code>__signbitf(GLIBC_2.2)</code> [ISO C99] |
| <code>__signbitl(GLIBC_2.2)</code> [ISO C99] | <code>acos(GLIBC_2.2)</code> [SUSv3] | <code>acosf(GLIBC_2.2)</code> [SUSv3] | <code>acosh(GLIBC_2.2)</code> [SUSv3] |
| <code>acoshf(GLIBC_2.2)</code> [SUSv3] | <code>acoshl(GLIBC_2.2)</code> [SUSv3] | <code>acosl(GLIBC_2.2)</code> [SUSv3] | <code>asin(GLIBC_2.2)</code> [SUSv3] |
| <code>asinf(GLIBC_2.2)</code> [SUSv3] | <code>asinh(GLIBC_2.2)</code> [SUSv3] | <code>asinhf(GLIBC_2.2)</code> [SUSv3] | <code>asinhf(GLIBC_2.2)</code> [SUSv3] |
| <code>asinl(GLIBC_2.2)</code> [SUSv3] | <code>atan(GLIBC_2.2)</code> [SUSv3] | <code>atan2(GLIBC_2.2)</code> [SUSv3] | <code>atan2f(GLIBC_2.2)</code> [SUSv3] |
| <code>atan2l(GLIBC_2.2)</code> [SUSv3] | <code>atanf(GLIBC_2.2)</code> [SUSv3] | <code>atanh(GLIBC_2.2)</code> [SUSv3] | <code>atanhf(GLIBC_2.2)</code> [SUSv3] |
| <code>atanhl(GLIBC_2.2)</code> | <code>atanl(GLIBC_2.2)</code> | <code>cabs(GLIBC_2.2)</code> | <code>cabsf(GLIBC_2.2)</code> |

| | | | |
|----------------------------|-----------------------------|------------------------------|------------------------------|
|) [SUSv3] | [SUSv3] | [SUSv3] | [SUSv3] |
| cabsf(GLIBC_2.2) [SUSv3] | cacos(GLIBC_2.2) [SUSv3] | cacosf(GLIBC_2.2) [SUSv3] | cacosh(GLIBC_2.2) [SUSv3] |
| cacoshf(GLIBC_2.2) [SUSv3] | cacoshl(GLIBC_2.2) [SUSv3] | cacosl(GLIBC_2.2) [SUSv3] | carg(GLIBC_2.2) [SUSv3] |
| cargf(GLIBC_2.2) [SUSv3] | cargl(GLIBC_2.2) [SUSv3] | casin(GLIBC_2.2) [SUSv3] | casinf(GLIBC_2.2) [SUSv3] |
| casinh(GLIBC_2.2) [SUSv3] | casinhf(GLIBC_2.2) [SUSv3] | casinhl(GLIBC_2.2) [SUSv3] | casinl(GLIBC_2.2) [SUSv3] |
| catan(GLIBC_2.2) [SUSv3] | catanf(GLIBC_2.2) [SUSv3] | catanh(GLIBC_2.2) [SUSv3] | catanhf(GLIBC_2.2) [SUSv3] |
| catanhl(GLIBC_2.2) [SUSv3] | catanl(GLIBC_2.2) [SUSv3] | cbirt(GLIBC_2.2) [SUSv3] | cbirtf(GLIBC_2.2) [SUSv3] |
| cbirtl(GLIBC_2.2) [SUSv3] | ccos(GLIBC_2.2) [SUSv3] | ccosf(GLIBC_2.2) [SUSv3] | ccosh(GLIBC_2.2) [SUSv3] |
| ccoshf(GLIBC_2.2) [SUSv3] | ccoshl(GLIBC_2.2) [SUSv3] | ccosl(GLIBC_2.2) [SUSv3] | ceil(GLIBC_2.2) [SUSv3] |
| ceilf(GLIBC_2.2) [SUSv3] | ceill(GLIBC_2.2) [SUSv3] | cexp(GLIBC_2.2) [SUSv3] | cexpf(GLIBC_2.2) [SUSv3] |
| cexpl(GLIBC_2.2) [SUSv3] | cimag(GLIBC_2.2) [SUSv3] | cimagf(GLIBC_2.2) [SUSv3] | cimagl(GLIBC_2.2) [SUSv3] |
| clog(GLIBC_2.2) [SUSv3] | clog10(GLIBC_2.2) [ISOC99] | clog10f(GLIBC_2.2) [ISOC99] | clog10l(GLIBC_2.2) [ISOC99] |
| clogf(GLIBC_2.2) [SUSv3] | clogl(GLIBC_2.2) [SUSv3] | conj(GLIBC_2.2) [SUSv3] | conjf(GLIBC_2.2) [SUSv3] |
| conjl(GLIBC_2.2) [SUSv3] | copysign(GLIBC_2.2) [SUSv3] | copysignf(GLIBC_2.2) [SUSv3] | copysignl(GLIBC_2.2) [SUSv3] |
| cos(GLIBC_2.2) [SUSv3] | cosf(GLIBC_2.2) [SUSv3] | cosh(GLIBC_2.2) [SUSv3] | coshf(GLIBC_2.2) [SUSv3] |
| coshl(GLIBC_2.2) [SUSv3] | cosl(GLIBC_2.2) [SUSv3] | cpow(GLIBC_2.2) [SUSv3] | cpowf(GLIBC_2.2) [SUSv3] |
| cpowl(GLIBC_2.2) [SUSv3] | cproj(GLIBC_2.2) [SUSv3] | cprojf(GLIBC_2.2) [SUSv3] | cprojl(GLIBC_2.2) [SUSv3] |
| creal(GLIBC_2.2) [SUSv3] | crealf(GLIBC_2.2) [SUSv3] | creall(GLIBC_2.2) [SUSv3] | csin(GLIBC_2.2) [SUSv3] |
| csinf(GLIBC_2.2) [SUSv3] | csinh(GLIBC_2.2) [SUSv3] | csinhf(GLIBC_2.2) [SUSv3] | csinhl(GLIBC_2.2) [SUSv3] |
| csinl(GLIBC_2.2) [SUSv3] | csqrt(GLIBC_2.2) [SUSv3] | csqrtf(GLIBC_2.2) [SUSv3] | csqrtl(GLIBC_2.2) [SUSv3] |
| ctan(GLIBC_2.2) [SUSv3] | ctanf(GLIBC_2.2) [SUSv3] | ctanh(GLIBC_2.2) [SUSv3] | ctanhf(GLIBC_2.2) [SUSv3] |

| | | | |
|---------------------------------------|-------------------------------------|-------------------------------------|---------------------------------------|
| ctanh1(GLIBC_2.2) [SUSv3] | ctanh(GLIBC_2.2) [SUSv3] | dremf(GLIBC_2.2) [ISOC99] | drem1(GLIBC_2.2) [ISOC99] |
| erf(GLIBC_2.2) [SUSv3] | erfc(GLIBC_2.2) [SUSv3] | erfcf(GLIBC_2.2) [SUSv3] | erfcl(GLIBC_2.2) [SUSv3] |
| erff(GLIBC_2.2) [SUSv3] | erfl(GLIBC_2.2) [SUSv3] | exp(GLIBC_2.2) [SUSv3] | exp2(GLIBC_2.2) [SUSv3] |
| exp2f(GLIBC_2.2) [SUSv3] | exp2l(GLIBC_2.2) [SUSv3] | expf(GLIBC_2.2) [SUSv3] | expl(GLIBC_2.2) [SUSv3] |
| expm1(GLIBC_2.2) [SUSv3] | expm1f(GLIBC_2.2) [SUSv3] | expm1l(GLIBC_2.2) [SUSv3] | fabs(GLIBC_2.2) [SUSv3] |
| fabsf(GLIBC_2.2) [SUSv3] | fabsl(GLIBC_2.2) [SUSv3] | fdim(GLIBC_2.2) [SUSv3] | fdimf(GLIBC_2.2) [SUSv3] |
| fdiml(GLIBC_2.2) [SUSv3] | feclearexcept(GLIBC_2.2) [SUSv3] | fegetenv(GLIBC_2.2) [SUSv3] | fegetexceptflag(GLIBC_2.2) [SUSv3] |
| fegetround(GLIBC_2.2) [SUSv3] | fehldexcept(GLIBC_2.2) [SUSv3] | feraiseexcept(GLIBC_2.2) [SUSv3] | fesetenv(GLIBC_2.2) [SUSv3] |
| fesetexceptflag(GLIBC_2.2) [SUSv3] | fesetround(GLIBC_2.2) [SUSv3] | fetestexcept(GLIBC_2.2) [SUSv3] | feupdateenv(GLIBC_2.2) [SUSv3] |
| finite(GLIBC_2.2) [SUSv2] | finitef(GLIBC_2.2) [ISOC99] | finitel(GLIBC_2.2) [ISOC99] | floor(GLIBC_2.2) [SUSv3] |
| floorf(GLIBC_2.2) [SUSv3] | floorl(GLIBC_2.2) [SUSv3] | fma(GLIBC_2.2) [SUSv3] | fmaf(GLIBC_2.2) [SUSv3] |
| fmal(GLIBC_2.2) [SUSv3] | fmax(GLIBC_2.2) [SUSv3] | fmaxf(GLIBC_2.2) [SUSv3] | fmaxl(GLIBC_2.2) [SUSv3] |
| fmin(GLIBC_2.2) [SUSv3] | fminf(GLIBC_2.2) [SUSv3] | fminl(GLIBC_2.2) [SUSv3] | fmod(GLIBC_2.2) [SUSv3] |
| fmodf(GLIBC_2.2) [SUSv3] | fmodl(GLIBC_2.2) [SUSv3] | frexp(GLIBC_2.2) [SUSv3] | frexpf(GLIBC_2.2) [SUSv3] |
| frexpl(GLIBC_2.2) [SUSv3] | gamma(GLIBC_2.2) [SUSv2] | gammaf(GLIBC_2.2) [ISOC99] | gammal(GLIBC_2.2) [ISOC99] |
| hypot(GLIBC_2.2) [SUSv3] | hypotf(GLIBC_2.2) [SUSv3] | hypotl(GLIBC_2.2) [SUSv3] | ilogb(GLIBC_2.2) [SUSv3] |
| ilogbf(GLIBC_2.2) [SUSv3] | ilogbl(GLIBC_2.2) [SUSv3] | j0(GLIBC_2.2) [SUSv3] | j0f(GLIBC_2.2) [ISOC99] |
| j0l(GLIBC_2.2) [ISOC99] | j1(GLIBC_2.2) [SUSv3] | j1f(GLIBC_2.2) [ISOC99] | j1l(GLIBC_2.2) [ISOC99] |
| jn(GLIBC_2.2) [SUSv3] | jnf(GLIBC_2.2) [ISOC99] | jnl(GLIBC_2.2) [ISOC99] | ldexp(GLIBC_2.2) [SUSv3] |
| ldexpf(GLIBC_2.2) [SUSv3] | ldexpl(GLIBC_2.2) [SUSv3] | lgamma(GLIBC_2.2) [SUSv3] | lgamma_r(GLIBC_2.2) [ISOC99] |
| lgammaf(GLIBC_2.2) [SUSv3] | lgammaf_r(GLIBC_2.2) [SUSv3] | lgammal(GLIBC_2.2) [SUSv3] | lgammal_r(GLIBC_2.2) [SUSv3] |

| | | | |
|-----------------------------------|--------------------------------|----------------------------------|-----------------------------------|
| 2.2) [SUSv3] | C_2.2) [ISOC99] | 2.2) [SUSv3] | _2.2) [ISOC99] |
| llrint(GLIBC_2.2) [SUSv3] | llrintf(GLIBC_2.2) [SUSv3] | llrintl(GLIBC_2.2) [SUSv3] | llround(GLIBC_2.2) [SUSv3] |
| llroundf(GLIBC_2.2) [SUSv3] | llroundl(GLIBC_2.2) [SUSv3] | log(GLIBC_2.2) [SUSv3] | log10(GLIBC_2.2) [SUSv3] |
| log10f(GLIBC_2.2) [SUSv3] | log10l(GLIBC_2.2) [SUSv3] | log1p(GLIBC_2.2) [SUSv3] | log1pf(GLIBC_2.2) [SUSv3] |
| log1pl(GLIBC_2.2) [SUSv3] | log2(GLIBC_2.2) [SUSv3] | log2f(GLIBC_2.2) [SUSv3] | log2l(GLIBC_2.2) [SUSv3] |
| logb(GLIBC_2.2) [SUSv3] | logbf(GLIBC_2.2) [SUSv3] | logbl(GLIBC_2.2) [SUSv3] | logf(GLIBC_2.2) [SUSv3] |
| logl(GLIBC_2.2) [SUSv3] | lrint(GLIBC_2.2) [SUSv3] | lrintf(GLIBC_2.2) [SUSv3] | lrintl(GLIBC_2.2) [SUSv3] |
| lround(GLIBC_2.2) [SUSv3] | lroundf(GLIBC_2.2) [SUSv3] | lroundl(GLIBC_2.2) [SUSv3] | matherr(GLIBC_2.2) [ISOC99] |
| modf(GLIBC_2.2) [SUSv3] | modff(GLIBC_2.2) [SUSv3] | modfl(GLIBC_2.2) [SUSv3] | nan(GLIBC_2.2) [SUSv3] |
| nanf(GLIBC_2.2) [SUSv3] | nanl(GLIBC_2.2) [SUSv3] | nearbyint(GLIBC_2.2) [SUSv3] | nearbyintf(GLIBC_2.2) [SUSv3] |
| nearbyintl(GLIBC_2.2) [SUSv3] | nextafter(GLIBC_2.2) [SUSv3] | nextafterf(GLIBC_2.2) [SUSv3] | nextafterl(GLIBC_2.2) [SUSv3] |
| nexttoward(GLIBC_2.2) [SUSv3] | nexttowardf(GLIBC_2.2) [SUSv3] | nexttowardl(GLIBC_2.2) [SUSv3] | pow(GLIBC_2.2) [SUSv3] |
| pow10(GLIBC_2.2) [ISOC99] | pow10f(GLIBC_2.2) [ISOC99] | pow10l(GLIBC_2.2) [ISOC99] | powf(GLIBC_2.2) [SUSv3] |
| powl(GLIBC_2.2) [SUSv3] | remainder(GLIBC_2.2) [SUSv3] | remainderf(GLIBC_2.2) [SUSv3] | remainderl(GLIBC_2.2) [SUSv3] |
| remquo(GLIBC_2.2) [SUSv3] | remquof(GLIBC_2.2) [SUSv3] | remquoL(GLIBC_2.2) [SUSv3] | rint(GLIBC_2.2) [SUSv3] |
| rintf(GLIBC_2.2) [SUSv3] | rintl(GLIBC_2.2) [SUSv3] | round(GLIBC_2.2) [SUSv3] | roundf(GLIBC_2.2) [SUSv3] |
| roundl(GLIBC_2.2) [SUSv3] | scalb(GLIBC_2.2) [SUSv3] | scalbf(GLIBC_2.2) [ISOC99] | scalbl(GLIBC_2.2) [ISOC99] |
| scalbln(GLIBC_2.2) [SUSv3] | scalblnf(GLIBC_2.2) [SUSv3] | scalblnl(GLIBC_2.2) [SUSv3] | scalbn(GLIBC_2.2) [SUSv3] |
| scalbnf(GLIBC_2.2) [SUSv3] | scalbnl(GLIBC_2.2) [SUSv3] | significantd(GLIBC_2.2) [ISOC99] | significantdf(GLIBC_2.2) [ISOC99] |
| significantdl(GLIBC_2.2) [ISOC99] | sin(GLIBC_2.2) [SUSv3] | sincos(GLIBC_2.2) [ISOC99] | sincosf(GLIBC_2.2) [ISOC99] |
| sincosl(GLIBC_2.2) [ISOC99] | sinf(GLIBC_2.2) [SUSv3] | sinh(GLIBC_2.2) [SUSv3] | sinhf(GLIBC_2.2) [SUSv3] |

| | | | |
|---------------------------------------|---------------------------------------|-------------------------------------|--------------------------------------|
| sinh(GLIBC_2.2) [SUSv3] | sinl(GLIBC_2.2) [SUSv3] | sqrt(GLIBC_2.2) [SUSv3] | sqrtf(GLIBC_2.2) [SUSv3] |
| sqrtl(GLIBC_2.2) [SUSv3] | tan(GLIBC_2.2) [SUSv3] | tanf(GLIBC_2.2) [SUSv3] | tanh(GLIBC_2.2) [SUSv3] |
| tanhf(GLIBC_2.2) [SUSv3] | tanhl(GLIBC_2.2) [SUSv3] | tanl(GLIBC_2.2) [SUSv3] | tgamma(GLIBC_2.2) [SUSv3] |
| tgammaf(GLIBC_2.2) [SUSv3] | tgammal(GLIBC_2.2) [SUSv3] | trunc(GLIBC_2.2) [SUSv3] | truncf(GLIBC_2.2) [SUSv3] |
| truncl(GLIBC_2.2) [SUSv3] | y0(GLIBC_2.2) [SUSv3] | y0f(GLIBC_2.2) [ISOC99] | y0l(GLIBC_2.2) [ISOC99] |
| y1(GLIBC_2.2) [SUSv3] | y1f(GLIBC_2.2) [ISOC99] | y1l(GLIBC_2.2) [ISOC99] | yn(GLIBC_2.2) [SUSv3] |
| ynf(GLIBC_2.2) [ISOC99] | ynl(GLIBC_2.2) [ISOC99] | | |

1898

1899

1900

1901

1902

1903

1904

1905

1906

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

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

~~[3]. this specification~~

~~[4]. SUSv2~~

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

1907

Table 11-26 libm - Math Data Interfaces

| | | | | |
|-----------------------------------|--|--|--|--|
| signgam(GLIBC_2.2) [1] | | | | |
|-----------------------------------|--|--|--|--|

1908

1909

1910

~~Referenced Specification(s)~~

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

| | | | |
|---------------------------------------|--|--|--|
| signgam(GLIBC_2.2) [SUSv3] | | | |
|---------------------------------------|--|--|--|

1911

11.5 Data Definitions for libm

1912

1913

1914

1915

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

1916

1917

1918

1919

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

1920

1921

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

1922 application developers should use this ABI to supplement - not to replace - source
 1923 interface definition specifications.

1924 This specification uses ~~ISO/IEC 9899~~the ISO C (1999) C Language as the reference
 1925 programming language, and data definitions are specified in ISO C format. The C
 1926 language is used here as a convenient notation. Using a C language description of
 1927 these data objects does not preclude their use by other programming languages.

11.5.1 complex.h

```

1928
1929 extern double cabs(double complex);
1930 extern float cabsf(float complex);
1931 extern long double cabsl(long double complex);
1932 extern double complex cacos(double complex);
1933 extern float complex cacosf(float complex);
1934 extern double complex cacosh(double complex);
1935 extern float complex cacoshf(float complex);
1936 extern long double complex cacoshl(long double complex);
1937 extern long double complex cacosl(long double complex);
1938 extern double carg(double complex);
1939 extern float cargf(float complex);
1940 extern long double cargl(long double complex);
1941 extern double complex casin(double complex);
1942 extern float complex casinf(float complex);
1943 extern double complex casinh(double complex);
1944 extern float complex casinhf(float complex);
1945 extern long double complex casinhl(long double complex);
1946 extern long double complex casinl(long double complex);
1947 extern double complex catan(double complex);
1948 extern float complex catanf(float complex);
1949 extern double complex catanh(double complex);
1950 extern float complex catanhf(float complex);
1951 extern long double complex catanhl(long double complex);
1952 extern long double complex catanl(long double complex);
1953 extern double complex ccos(double complex);
1954 extern float complex ccosf(float complex);
1955 extern double complex ccosh(double complex);
1956 extern float complex ccoshf(float complex);
1957 extern long double complex ccoshl(long double complex);
1958 extern long double complex ccosl(long double complex);
1959 extern double complex cexp(double complex);
1960 extern float complex cexpf(float complex);
1961 extern long double complex cexpl(long double complex);
1962 extern double cimag(double complex);
1963 extern float cimagf(float complex);
1964 extern long double cimagl(long double complex);
1965 extern double complex clog(double complex);
1966 extern float complex clog10f(float complex);
1967 extern long double complex clog10l(long double complex);
1968 extern float complex clogf(float complex);
1969 extern long double complex clogl(long double complex);
1970 extern double complex conj(double complex);
1971 extern float complex conjf(float complex);
1972 extern long double complex conjl(long double complex);
1973 extern double complex cpow(double complex, double complex);
1974 extern float complex cpowf(float complex, float complex);
1975 extern long double complex cpowl(long double complex, long double
1976 complex);
1977 extern double complex cproj(double complex);
1978 extern float complex cprojf(float complex);
1979 extern long double complex cprojl(long double complex);
1980 extern double creal(double complex);

```

```

1981 extern float crealf(float complex);
1982 extern long double creall(long double complex);
1983 extern double complex csin(double complex);
1984 extern float complex csinf(float complex);
1985 extern double complex csinh(double complex);
1986 extern float complex csinhf(float complex);
1987 extern long double complex csinhl(long double complex);
1988 extern long double complex csinl(long double complex);
1989 extern double complex csqrt(double complex);
1990 extern float complex csqrtf(float complex);
1991 extern long double complex csqrtl(long double complex);
1992 extern double complex ctan(double complex);
1993 extern float complex ctanf(float complex);
1994 extern double complex ctanh(double complex);
1995 extern float complex ctanhf(float complex);
1996 extern long double complex ctanhl(long double complex);
1997 extern long double complex ctanl(long double complex);

```

11.5.2 fenv.h

```

1998
1999 #define FE_INVALID      (1UL << 0)
2000 #define FE_DIVBYZERO   (1UL << 2)
2001 #define FE_OVERFLOW    (1UL << 3)
2002 #define FE_UNDERFLOW  (1UL << 4)
2003 #define FE_INEXACT     (1UL << 5)
2004 #define FE_UNNORMAL    1UL << 1
2005
2006 #define FE_ALL_EXCEPT \
2007     (FE_INEXACT | FE_UNDERFLOW | FE_OVERFLOW | FE_DIVBYZERO |
2008     FE_UNNORMAL | FE_INVALID)
2009
2010 #define FE_TONEAREST   0
2011 #define FE_DOWNWARD    1
2012 #define FE_UPWARD      2
2013 #define FE_TOWARDZERO  3
2014
2015 typedef unsigned long int fexcept_t;
2016
2017 typedef unsigned long int fenv_t;
2018
2019 #define FE_DFL_ENV      ((__const fenv_t *) 0xc009804c0270033FUL)
2020
2021 extern int feclearexcept(int);
2022 extern int fegetenv(fenv_t *);
2023 extern int fegetexceptflag(fexcept_t *, int);
2024 extern int fegetround(void);
2025 extern int feholdexcept(fenv_t *);
2026 extern int feraiseexcept(int);
2027 extern int fesetenv(const fenv_t *);
2028 extern int fesetexceptflag(const fexcept_t *, int);
2029 extern int fesetround(int);
2030 extern int fetestexcept(int);
2031 extern int feupdateenv(const fenv_t *);

```

11.5.23 math.h

```

2032
2033 #define fpclassify(x) \
2034     (sizeof (x) == sizeof (float) ? __fpclassifyf (x) : sizeof (x)
2035     == sizeof (double) ? __fpclassify (x) : __fpclassifyl (x))
2036 #define signbit(x) \
2037     (sizeof (x) == sizeof (float)? __signbitf (x) : sizeof (x) ==
2038     sizeof (double)? __signbit (x) : __signbitl (x))

```

```

2039
2040     #define FP_ILOGE0      -2147483648
2041     #define FP_ILOGBNAN   2147483647
2042
2043     extern int  __finite(double);
2044     extern int  __finitef(float);
2045     extern int  __finitel(long double);
2046     extern int  __isinf(double);
2047     extern int  __isinff(float);
2048     extern int  __isinfl(long double);
2049     extern int  __isnan(double);
2050     extern int  __isnanf(float);
2051     extern int  __isnanl(long double);
2052     extern int  __signbit(double);
2053     extern int  __signbitf(float);
2054     extern int  __fpclassify(double);
2055     extern int  __fpclassifyf(float);
2056     extern int  __fpclassifyl(long double);
2057     extern int  signgam(void);
2058     extern double copysign(double, double);
2059     extern int  finite(double);
2060     extern double frexp(double, int *);
2061     extern double ldexp(double, int);
2062     extern double modf(double, double *);
2063     extern double acos(double);
2064     extern double acosh(double);
2065     extern double asinh(double);
2066     extern double atanh(double);
2067     extern double asin(double);
2068     extern double atan(double);
2069     extern double atan2(double, double);
2070     extern double cbrt(double);
2071     extern double ceil(double);
2072     extern double cos(double);
2073     extern double cosh(double);
2074     extern double erf(double);
2075     extern double erfc(double);
2076     extern double exp(double);
2077     extern double expm1(double);
2078     extern double fabs(double);
2079     extern double floor(double);
2080     extern double fmod(double, double);
2081     extern double gamma(double);
2082     extern double hypot(double, double);
2083     extern int  ilogb(double);
2084     extern double j0(double);
2085     extern double j1(double);
2086     extern double jn(int, double);
2087     extern double lgamma(double);
2088     extern double log(double);
2089     extern double log10(double);
2090     extern double loglp(double);
2091     extern double logb(double);
2092     extern double nextafter(double, double);
2093     extern double pow(double, double);
2094     extern double remainder(double, double);
2095     extern double rint(double);
2096     extern double scalb(double, double);
2097     extern double sin(double);
2098     extern double sinh(double);
2099     extern double sqrt(double);
2100     extern double tan(double);
2101     extern double tanh(double);
2102     extern double y0(double);

```

```

2103     extern double yl(double);
2104     extern double yn(int, double);
2105     extern float copysignf(float, float);
2106     extern long double copysignl(long double, long double);
2107     extern int finitef(float);
2108     extern int finitel(long double);
2109     extern float frexpf(float, int *);
2110     extern long double frexpl(long double, int *);
2111     extern float ldexpf(float, int);
2112     extern long double ldexpl(long double, int);
2113     extern float modff(float, float *);
2114     extern long double modfl(long double, long double *);
2115     extern double scalbln(double, long int);
2116     extern float scalblnf(float, long int);
2117     extern long double scalblnl(long double, long int);
2118     extern double scalbn(double, int);
2119     extern float scalbnf(float, int);
2120     extern long double scalbnl(long double, int);
2121     extern float acosf(float);
2122     extern float acoshf(float);
2123     extern long double acoshl(long double);
2124     extern long double acosl(long double);
2125     extern float asinf(float);
2126     extern float asinhf(float);
2127     extern long double asinhl(long double);
2128     extern long double asinl(long double);
2129     extern float atan2f(float, float);
2130     extern long double atan2l(long double, long double);
2131     extern float atanf(float);
2132     extern float atanhf(float);
2133     extern long double atanhhl(long double);
2134     extern long double atanl(long double);
2135     extern float cbrtf(float);
2136     extern long double cbrtl(long double);
2137     extern float ceilf(float);
2138     extern long double ceill(long double);
2139     extern float cosf(float);
2140     extern float coshf(float);
2141     extern long double coshl(long double);
2142     extern long double cosl(long double);
2143     extern float dremf(float, float);
2144     extern long double dreml(long double, long double);
2145     extern float erfcf(float);
2146     extern long double erfcl(long double);
2147     extern float erff(float);
2148     extern long double erfl(long double);
2149     extern double exp2(double);
2150     extern float exp2f(float);
2151     extern long double exp2l(long double);
2152     extern float expf(float);
2153     extern long double expl(long double);
2154     extern float expmlf(float);
2155     extern long double expmll(long double);
2156     extern float fabsf(float);
2157     extern long double fabsl(long double);
2158     extern double fdim(double, double);
2159     extern float fdimf(float, float);
2160     extern long double fdiml(long double, long double);
2161     extern float floorf(float);
2162     extern long double floorl(long double);
2163     extern double fma(double, double, double);
2164     extern float fmaf(float, float, float);
2165     extern long double fmal(long double, long double, long double);
2166     extern double fmax(double, double);

```



```

2167 extern float fmaxf(float, float);
2168 extern long double fmaxl(long double, long double);
2169 extern double fmin(double, double);
2170 extern float fminf(float, float);
2171 extern long double fminl(long double, long double);
2172 extern float fmodf(float, float);
2173 extern long double fmodl(long double, long double);
2174 extern float gammaf(float);
2175 extern long double gammal(long double);
2176 extern float hypotf(float, float);
2177 extern long double hypotl(long double, long double);
2178 extern int ilogbf(float);
2179 extern int ilogbl(long double);
2180 extern float j0f(float);
2181 extern long double j0l(long double);
2182 extern float j1f(float);
2183 extern long double j1l(long double);
2184 extern float jnf(int, float);
2185 extern long double jnl(int, long double);
2186 extern double lgamma_r(double, int *);
2187 extern float lgammaf(float);
2188 extern float lgammaf_r(float, int *);
2189 extern long double lgammal(long double);
2190 extern long double lgammal_r(long double, int *);
2191 extern long int llrint(double);
2192 extern long long int llrintf(float);
2193 extern long long int llrintl(long double);
2194 extern long long int llround(double);
2195 extern long long int llroundf(float);
2196 extern long long int llroundl(long double);
2197 extern float log10f(float);
2198 extern long double log10l(long double);
2199 extern float log1pf(float);
2200 extern long double log1pl(long double);
2201 extern double log2(double);
2202 extern float log2f(float);
2203 extern long double log2l(long double);
2204 extern float logbf(float);
2205 extern long double logbl(long double);
2206 extern float logf(float);
2207 extern long double logl(long double);
2208 extern long int lrint(double);
2209 extern long int lrintf(float);
2210 extern long int lrintl(long double);
2211 extern long int lround(double);
2212 extern long int lroundf(float);
2213 extern long int lroundl(long double);
2214 extern int matherr(struct exception *);
2215 extern double nan(const char *);
2216 extern float nanf(const char *);
2217 extern long double nanl(const char *);
2218 extern double nearbyint(double);
2219 extern float nearbyintf(float);
2220 extern long double nearbyintl(long double);
2221 extern float nextafterf(float, float);
2222 extern long double nextafterl(long double, long double);
2223 extern double nexttoward(double, long double);
2224 extern float nexttowardf(float, long double);
2225 extern long double nexttowardl(long double, long double);
2226 extern double pow10(double);
2227 extern float pow10f(float);
2228 extern long double pow10l(long double);
2229 extern float powf(float, float);
2230 extern long double powl(long double, long double);

```

```

2231     extern float remainderf(float, float);
2232     extern long double remainderl(long double, long double);
2233     extern double remquo(double, double, int *);
2234     extern float remquof(float, float, int *);
2235     extern long double remquo1(long double, long double, int *);
2236     extern float rintf(float);
2237     extern long double rintl(long double);
2238     extern double round(double);
2239     extern float roundf(float);
2240     extern long double roundl(long double);
2241     extern float scalbf(float, float);
2242     extern long double scalbl(long double, long double);
2243     extern double significand(double);
2244     extern float significandf(float);
2245     extern long double significandl(long double);
2246     extern void sincos(double, double *, double *);
2247     extern void sincosf(float, float *, float *);
2248     extern void sincosl(long double, long double *, long double *);
2249     extern float sinf(float);
2250     extern float sinhf(float);
2251     extern long double sinhl(long double);
2252     extern long double sinl(long double);
2253     extern float sqrtf(float);
2254     extern long double sqrtl(long double);
2255     extern float tanf(float);
2256     extern float tanhf(float);
2257     extern long double tanhl(long double);
2258     extern long double tanl(long double);
2259     extern double tgamma(double);
2260     extern float tgammaf(float);
2261     extern long double tgammal(long double);
2262     extern double trunc(double);
2263     extern float truncf(float);
2264     extern long double trunc1(long double);
2265     extern float y0f(float);
2266     extern long double y0l(long double);
2267     extern float y1f(float);
2268     extern long double y1l(long double);
2269     extern float ynf(int, float);
2270     extern long double ynl(int, long double);
2271     extern int __fpclassifyl(long double);
2272     extern int __fpclassifyl(long double);
2273     extern int __signbitl(long double);
2274     extern int __signbitl(long double);
2275     extern int __signbitl(long double);
2276     extern long double exp2l(long double);
2277     extern long double exp2l(long double);

```

11.6 Interfaces for libpthread

2278 Table 11-27 defines the library name and shared object name for the libpthread
 2279 library

2280 **Table 11-27 libpthread Definition**

| | |
|----------|-----------------|
| Library: | libpthread |
| SONAME: | libpthread.so.0 |

2281

2282 The behavior of the interfaces in this library is specified by the following specifica-
 2283 tions:

[LFS] Large File Support

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

11.6.1 Realtime Threads

11.6.1.1 Interfaces for Realtime Threads

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

Table 11-28 libpthread - Realtime Threads Function Interfaces

| | | | | |
|--|--|---|--|--|
| pthread_attr_ getinheritsche d(GLIBC_2.2) [1] | pthread_attr_ getscope(GLI BC_2.2)[1] | pthread_attr_ setschedpolie y(GLIBC_2.2) [1] | pthread_getse hedparam(GL IBC_2.2)[1] | |
| pthread_attr_ getschedpolie y(GLIBC_2.2) [1] | pthread_attr_ setinheritsche d(GLIBC_2.2) [1] | pthread_attr_ setscope(GLI BC_2.2)[1] | pthread_setse hedparam(GL IBC_2.2)[1] | |

Referenced Specification(s)

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

| | | | |
|--|---|---|--|
| pthread_attr_geti nheritsched(GLIB C_2.2) [SUSv3] | pthread_attr_gets chedpolicy(GLIB C_2.2) [SUSv3] | pthread_attr_gets cope(GLIBC_2.2) [SUSv3] | pthread_attr_setin heritsched(GLIBC _2.2) [SUSv3] |
| pthread_attr_setsc hedpolicy(GLIBC _2.2) [SUSv3] | pthread_attr_setsc ope(GLIBC_2.2) [SUSv3] | pthread_getsched param(GLIBC_2.2) [SUSv3] | pthread_setsched param(GLIBC_2.2) [SUSv3] |

11.6.2 Advanced Realtime Threads

11.6.2.1 Interfaces for Advanced Realtime Threads

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

11.6.3 Posix Threads

11.6.3.1 Interfaces for Posix Threads

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

Table 11-29 libpthread - Posix Threads Function Interfaces

| | | | | |
|---|--|---|---|---|
| _pthread_clea nup_pop(GLI BC_2.2)[1] | pthread_cond _broadcast(G LIBC_2.3.2) [2] | pthread_join(GLIBC_2.2) [2] | pthread_rwlock ek_destroy(G LIBC_2.2)[2] | pthread_seteo neurrency(GL IBC_2.2)[2] |
| _pthread_clea | pthread_cond | pthread_key_ | pthread_rwlock | pthread_setsp |

| | | | | |
|--|---|---|--|-----------------------------------|
| nup_push(GLIBC_2.2) [1] | _destroy(GLIBC_2.3.2) [2] | create(GLIBC_2.2) [2] | ek_init(GLIBC_2.2) [2] | ecific(GLIBC_2.2) [2] |
| pthread_attr_destroy(GLIBC_2.2) [2] | pthread_cond_init(GLIBC_2.3.2) [2] | pthread_key_delete(GLIBC_2.2) [2] | pthread_rwlock_rdlock(GLIBC_2.2) [2] | pthread_sigmask(GLIBC_2.2) [2] |
| pthread_attr_getdetachstate(GLIBC_2.2) [2] | pthread_cond_signal(GLIBC_2.3.2) [2] | pthread_kill(GLIBC_2.2) [2] | pthread_rwlock_timedrdlock(GLIBC_2.2) [2] | pthread_testcancel(GLIBC_2.2) [2] |
| pthread_attr_getguardsize(GLIBC_2.2) [2] | pthread_cond_timedwait(GLIBC_2.3.2) [2] | pthread_mutex_destroy(GLIBC_2.2) [2] | pthread_rwlock_timedwrlock(GLIBC_2.2) [2] | sem_close(GLIBC_2.2) [2] |
| pthread_attr_getschedparam(GLIBC_2.2) [2] | pthread_cond_wait(GLIBC_2.3.2) [2] | pthread_mutex_init(GLIBC_2.2) [2] | pthread_rwlock_tryrdlock(GLIBC_2.2) [2] | sem_destroy(GLIBC_2.2) [2] |
| pthread_attr_getstack(GLIBC_2.2) [2] | pthread_cond_attr_destroy(GLIBC_2.2) [2] | pthread_mutex_lock(GLIBC_2.2) [2] | pthread_rwlock_trywrlock(GLIBC_2.2) [2] | sem_getvalue(GLIBC_2.2) [2] |
| pthread_attr_getstackaddr(GLIBC_2.2) [2] | pthread_cond_attr_getpshared(GLIBC_2.2) [2] | pthread_mutex_trylock(GLIBC_2.2) [2] | pthread_rwlock_unlock(GLIBC_2.2) [2] | sem_init(GLIBC_2.2) [2] |
| pthread_attr_getstacksize(GLIBC_2.2) [2] | pthread_cond_attr_init(GLIBC_2.2) [2] | pthread_mutex_unlock(GLIBC_2.2) [2] | pthread_rwlock_wrlock(GLIBC_2.2) [2] | sem_open(GLIBC_2.2) [2] |
| pthread_attr_init(GLIBC_2.2) [2] | pthread_cond_attr_setpshared(GLIBC_2.2) [2] | pthread_mutexattr_destroy(GLIBC_2.2) [2] | pthread_rwlockattr_destroy(GLIBC_2.2) [2] | sem_post(GLIBC_2.2) [2] |
| pthread_attr_setdetachstate(GLIBC_2.2) [2] | pthread_create(GLIBC_2.2) [2] | pthread_mutexattr_getpshared(GLIBC_2.2) [2] | pthread_rwlockattr_getpshared(GLIBC_2.2) [2] | sem_timedwait(GLIBC_2.2) [2] |
| pthread_attr_setguardsize(GLIBC_2.2) [2] | pthread_detach(GLIBC_2.2) [2] | pthread_mutexattr_gettype(GLIBC_2.2) [2] | pthread_rwlockattr_init(GLIBC_2.2) [2] | sem_trywait(GLIBC_2.2) [2] |
| pthread_attr_setschedparam(GLIBC_2.2) [2] | pthread_equal(GLIBC_2.2) [2] | pthread_mutexattr_init(GLIBC_2.2) [2] | pthread_rwlockattr_setpshared(GLIBC_2.2) [2] | sem_unlink(GLIBC_2.2) [2] |
| pthread_attr_ | pthread_exit(| pthread_mutex | pthread_self(| sem_wait(GLI |

| | | | | |
|---|--|---|---|-------------|
| setstackaddr(GLIBC_2.2) [2] | GLIBC_2.2) [2] | xattr_setpshar ed(GLIBC_2.2) [2] | GLIBC_2.2) [2] | BC_2.2) [2] |
| pthread_attr_ setstacksize(G LIBC_2.3.3) [2] | pthread_gete ncurrency(G LIBC_2.2) [2] | pthread_mute xattr_settype(GLIBC_2.2) [2] | pthread_setea ncelstate(GLI BC_2.2) [2] | |
| pthread_cane el(GLIBC_2.2) [2] | pthread_gets pecific(GLIBC _2.2) [2] | pthread_once (GLIBC_2.2) [2] | pthread_setea nceltype(GLI BC_2.2) [2] | |

2302

2303

Referenced Specification(s)

2304

[1]. this specification

2305

[2]. ISO POSIX (2003)

| | | | |
|---|--|--|--|
| _pthread_cleanup _pop(GLIBC_2.2) [LSB] | _pthread_cleanup _push(GLIBC_2.2) [LSB] | pthread_attr_dest roy(GLIBC_2.2) [SUSv3] | pthread_attr_getd etachstate(GLIBC _2.2) [SUSv3] |
| pthread_attr_getg uardsize(GLIBC_2 .2) [SUSv3] | pthread_attr_gets chedparam(GLIB C_2.2) [SUSv3] | pthread_attr_getst ack(GLIBC_2.2) [SUSv3] | pthread_attr_getst ackaddr(GLIBC_2 .2) [SUSv3] |
| pthread_attr_getst acksize(GLIBC_2. 2) [SUSv3] | pthread_attr_init(GLIBC_2.2) [SUSv3] | pthread_attr_setd etachstate(GLIBC _2.2) [SUSv3] | pthread_attr_setg uardsize(GLIBC_2 .2) [SUSv3] |
| pthread_attr_setsc hedparam(GLIBC _2.2) [SUSv3] | pthread_attr_setst ackaddr(GLIBC_2 .2) [SUSv3] | pthread_attr_setst acksize(GLIBC_2. 3.3) [SUSv3] | pthread_cancel(G LIBC_2.2) [SUSv3] |
| pthread_cond_bro adcast(GLIBC_2.3. 2) [SUSv3] | pthread_cond_des troy(GLIBC_2.3.2) [SUSv3] | pthread_cond_init (GLIBC_2.3.2) [SUSv3] | pthread_cond_sig nal(GLIBC_2.3.2) [SUSv3] |
| pthread_cond_tim edwait(GLIBC_2.3 .2) [SUSv3] | pthread_cond_wa it(GLIBC_2.3.2) [SUSv3] | pthread_condattr _destroy(GLIBC_ 2.2) [SUSv3] | pthread_condattr _getpshared(GLIB C_2.2) [SUSv3] |
| pthread_condattr _init(GLIBC_2.2) [SUSv3] | pthread_condattr _setpshared(GLIB C_2.2) [SUSv3] | pthread_create(G LIBC_2.2) [SUSv3] | pthread_detach(G LIBC_2.2) [SUSv3] |
| pthread_equal(GL IBC_2.2) [SUSv3] | pthread_exit(GLI BC_2.2) [SUSv3] | pthread_getconcu rrency(GLIBC_2.2) [SUSv3] | pthread_getspecif ic(GLIBC_2.2) [SUSv3] |
| pthread_join(GLI BC_2.2) [SUSv3] | pthread_key_crea te(GLIBC_2.2) [SUSv3] | pthread_key_dele te(GLIBC_2.2) [SUSv3] | pthread_kill(GLIB C_2.2) [SUSv3] |
| pthread_mutex_d estroy(GLIBC_2.2) [SUSv3] | pthread_mutex_in it(GLIBC_2.2) [SUSv3] | pthread_mutex_lo ck(GLIBC_2.2) [SUSv3] | pthread_mutex_tr ylock(GLIBC_2.2) [SUSv3] |

| | | | |
|--|---|--|---|
| pthread_mutex_unlock(GLIBC_2.2) [SUSv3] | pthread_mutexattr_destroy(GLIBC_2.2) [SUSv3] | pthread_mutexattr_getpshared(GLIBC_2.2) [SUSv3] | pthread_mutexattr_gettype(GLIBC_2.2) [SUSv3] |
| pthread_mutexattr_init(GLIBC_2.2) [SUSv3] | pthread_mutexattr_setpshared(GLIBC_2.2) [SUSv3] | pthread_mutexattr_settype(GLIBC_2.2) [SUSv3] | pthread_once(GLIBC_2.2) [SUSv3] |
| pthread_rwlock_destroy(GLIBC_2.2) [SUSv3] | pthread_rwlock_init(GLIBC_2.2) [SUSv3] | pthread_rwlock_rdlock(GLIBC_2.2) [SUSv3] | pthread_rwlock_timedrdlock(GLIBC_2.2) [SUSv3] |
| pthread_rwlock_timedwrlock(GLIBC_2.2) [SUSv3] | pthread_rwlock_tryrdlock(GLIBC_2.2) [SUSv3] | pthread_rwlock_trywrlock(GLIBC_2.2) [SUSv3] | pthread_rwlock_unlock(GLIBC_2.2) [SUSv3] |
| pthread_rwlock_wrlock(GLIBC_2.2) [SUSv3] | pthread_rwlockattr_destroy(GLIBC_2.2) [SUSv3] | pthread_rwlockattr_getpshared(GLIBC_2.2) [SUSv3] | pthread_rwlockattr_init(GLIBC_2.2) [SUSv3] |
| pthread_rwlockattr_setpshared(GLIBC_2.2) [SUSv3] | pthread_self(GLIBC_2.2) [SUSv3] | pthread_setcancelstate(GLIBC_2.2) [SUSv3] | pthread_setcanceltype(GLIBC_2.2) [SUSv3] |
| pthread_setconcurrency(GLIBC_2.2) [SUSv3] | pthread_setspecific(GLIBC_2.2) [SUSv3] | pthread_sigmask(GLIBC_2.2) [SUSv3] | pthread_testcancel(GLIBC_2.2) [SUSv3] |
| sem_close(GLIBC_2.2) [SUSv3] | sem_destroy(GLIBC_2.2) [SUSv3] | sem_getvalue(GLIBC_2.2) [SUSv3] | sem_init(GLIBC_2.2) [SUSv3] |
| sem_open(GLIBC_2.2) [SUSv3] | sem_post(GLIBC_2.2) [SUSv3] | sem_timedwait(GLIBC_2.2) [SUSv3] | sem_trywait(GLIBC_2.2) [SUSv3] |
| sem_unlink(GLIBC_2.2) [SUSv3] | sem_wait(GLIBC_2.2) [SUSv3] | | |

2306

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

2307

2308

2309

2310

2311

2312

2313

2314

2315

Referenced Specification(s)

[1]:

| | | | |
|------------------|-----------------|------------------|------------------|
| lseek64(GLIBC_2. | open64(GLIBC_2. | pread(GLIBC_2.2) | pread64(GLIBC_2. |
|------------------|-----------------|------------------|------------------|

| | | | |
|---------------------------|---------------------------|---------|----------|
| 2) [LFS] | 2) [LFS] | [SUSv3] | 2) [LFS] |
| pwrite(GLIBC_2.2) [SUSv3] | pwrite64(GLIBC_2.2) [LFS] | | |

2316

11.7 Data Definitions for libpthread

2317

2318

2319

2320

2321

2322

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.

2323

2324

2325

2326

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.

2327

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

2328

2329

2330

2331

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

11.7.1 pthread.h

2332

2333

2334

2335

2336

2337

2338

2339

2340

2341

2342

2343

2344

2345

2346

2347

2348

2349

2350

2351

2352

2353

2354

2355

2356

2357

2358

2359

2360

2361

2362

2363

2364

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

```

2365         int __detachstate;
2366         int __schedpolicy;
2367         struct sched_param
2368     __schedparam;
2369         int __inheritsched;
2370         int __scope;
2371         size_t __guardsize;
2372         int __stackaddr_set;
2373         void *__stackaddr;
2374         unsigned long int __stacksize;}
2375     pthread_attr_t *, struct
2376     sched_param {
2377         int sched_priority;}
2378
2379     *);
2380     extern int pthread_attr_getschedpolicy(const typedef struct {
2381         int __detachstate;
2382         int __schedpolicy;
2383         struct sched_param
2384     __schedparam;
2385         int __inheritsched;
2386         int __scope;
2387         size_t __guardsize;
2388         int __stackaddr_set;
2389         void *__stackaddr;
2390         unsigned long int __stacksize;}
2391     pthread_attr_t *, int *);
2392     extern int pthread_attr_getscope(const typedef struct {
2393         int __detachstate;
2394         int __schedpolicy;
2395         struct sched_param __schedparam;
2396         int __inheritsched;
2397         int __scope;
2398         size_t __guardsize;
2399         int __stackaddr_set;
2400         void *__stackaddr;
2401         unsigned long int __stacksize;}
2402     pthread_attr_t *, int *);
2403     extern int pthread_attr_init(pthread_attr_t *);
2404     extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
2405     extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
2406     extern int pthread_attr_setschedparam(pthread_attr_t *, const struct
2407     sched_param {
2408         int sched_priority;}
2409
2410     *);
2411     extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
2412     extern int pthread_attr_setscope(pthread_attr_t *, int);
2413     extern int pthread_cancel(typedef unsigned long int pthread_t);
2414     extern int pthread_cond_broadcast(pthread_cond_t *);
2415     extern int pthread_cond_destroy(pthread_cond_t *);
2416     extern int pthread_cond_init(pthread_cond_t *, const typedef struct {
2417         int __dummy;}
2418
2419         pthread_condattr_t *);
2420     extern int pthread_cond_signal(pthread_cond_t *);
2421     extern int pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
2422     const struct timespec {
2423         time_t tv_sec; long int tv_nsec;}
2424
2425     *);
2426     extern int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
2427     extern int pthread_condattr_destroy(pthread_condattr_t *);
2428     extern int pthread_condattr_init(pthread_condattr_t *);

```



```

2429 extern int pthread_create(pthread_t *, const typedef struct {
2430     int __detachstate;
2431     int __schedpolicy;
2432     struct sched_param __schedparam;
2433     int __inheritsched;
2434     int __scope;
2435     size_t __guardsize;
2436     int __stackaddr_set;
2437     void *__stackaddr;
2438     unsigned long int __stacksize;}
2439     pthread_attr_t *,
2440     void *(*__start_routine) (void *p1)
2441     , void *);
2442 extern int pthread_detach(typedef unsigned long int pthread_t);
2443 extern int pthread_equal(typedef unsigned long int pthread_t,
2444     typedef unsigned long int pthread_t);
2445 extern void pthread_exit(void *);
2446 extern int pthread_getschedparam(typedef unsigned long int pthread_t,
2447     int *, struct sched_param {
2448     int sched_priority;}
2449
2450     *);
2451 extern void *pthread_getspecific(typedef unsigned int pthread_key_t);
2452 extern int pthread_join(typedef unsigned long int pthread_t, void **);
2453 extern int pthread_key_create(pthread_key_t *, void (*destr_func) (void
2454 *)
2455     );
2456 extern int pthread_key_delete(typedef unsigned int pthread_key_t);
2457 extern int pthread_mutex_destroy(pthread_mutex_t *);
2458 extern int pthread_mutex_init(pthread_mutex_t *, const typedef struct
2459 {
2460     int __mutexkind;}
2461
2462     pthread_mutexattr_t *);
2463 extern int pthread_mutex_lock(pthread_mutex_t *);
2464 extern int pthread_mutex_trylock(pthread_mutex_t *);
2465 extern int pthread_mutex_unlock(pthread_mutex_t *);
2466 extern int pthread_mutexattr_destroy(pthread_mutexattr_t *);
2467 extern int pthread_mutexattr_init(pthread_mutexattr_t *);
2468 extern int pthread_once(pthread_once_t *, void (*init_routine) (void)
2469     );
2470 extern int pthread_rwlock_destroy(pthread_rwlock_t *);
2471 extern int pthread_rwlock_init(pthread_rwlock_t *,
2472     pthread_rwlockattr_t *);
2473 extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
2474 extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *);
2475 extern int pthread_rwlock_trywrlock(pthread_rwlock_t *);
2476 extern int pthread_rwlock_unlock(pthread_rwlock_t *);
2477 extern int pthread_rwlock_wrlock(pthread_rwlock_t *);
2478 extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
2479 extern int pthread_rwlockattr_getpshared(const typedef struct {
2480     int __lockkind; int
2481     __pshared;}
2482     pthread_rwlockattr_t *, int
2483     *);
2484 extern int pthread_rwlockattr_init(pthread_rwlockattr_t *);
2485 extern int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
2486 extern typedef unsigned long int pthread_t pthread_self(void);
2487 extern int pthread_setcancelstate(int, int *);
2488 extern int pthread_setcanceltype(int, int *);
2489 extern int pthread_setschedparam(typedef unsigned long int pthread_t,
2490     int, const struct sched_param {
2491     int sched_priority;}
2492

```

```

2493         *);
2494 extern int pthread_setspecific(typedef unsigned int pthread_key_t,
2495                               const void *);
2496 extern void pthread_testcancel(void);
2497 extern int pthread_attr_getguardsize(const typedef struct {
2498                                     int __detachstate;
2499                                     int __schedpolicy;
2500                                     struct sched_param __schedparam;
2501                                     int __inheritsched;
2502                                     int __scope;
2503                                     size_t __guardsize;
2504                                     int __stackaddr_set;
2505                                     void *__stackaddr;
2506                                     unsigned long int __stacksize;}
2507                                     pthread_attr_t *, size_t *);
2508 extern int pthread_attr_setguardsize(pthread_attr_t *,
2509                                     typedef unsigned long int
2510                                     size_t);
2511 extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
2512 extern int pthread_attr_getstackaddr(const typedef struct {
2513                                     int __detachstate;
2514                                     int __schedpolicy;
2515                                     struct sched_param __schedparam;
2516                                     int __inheritsched;
2517                                     int __scope;
2518                                     size_t __guardsize;
2519                                     int __stackaddr_set;
2520                                     void *__stackaddr;
2521                                     unsigned long int __stacksize;}
2522                                     pthread_attr_t *, void **);
2523 extern int pthread_attr_setstacksize(pthread_attr_t *,
2524                                     typedef unsigned long int
2525                                     size_t);
2526 extern int pthread_attr_getstacksize(const typedef struct {
2527                                     int __detachstate;
2528                                     int __schedpolicy;
2529                                     struct sched_param __schedparam;
2530                                     int __inheritsched;
2531                                     int __scope;
2532                                     size_t __guardsize;
2533                                     int __stackaddr_set;
2534                                     void *__stackaddr;
2535                                     unsigned long int __stacksize;}
2536                                     pthread_attr_t *, size_t *);
2537 extern int pthread_mutexattr_gettype(const typedef struct {
2538                                     int __mutexkind;}
2539                                     pthread_mutexattr_t *, int *);
2540 extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
2541 extern int pthread_getconcurrency(void);
2542 extern int pthread_setconcurrency(int);
2543 extern int pthread_attr_getstack(const typedef struct {
2544                                     int __detachstate;
2545                                     int __schedpolicy;
2546                                     struct sched_param __schedparam;
2547                                     int __inheritsched;
2548                                     int __scope;
2549                                     size_t __guardsize;
2550                                     int __stackaddr_set;
2551                                     void *__stackaddr;
2552                                     unsigned long int __stacksize;}
2553                                     pthread_attr_t *, void **, size_t *);
2554 extern int pthread_attr_setstack(pthread_attr_t *, void *,
2555                                     typedef unsigned long int size_t);
2556 extern int pthread_condattr_getpshared(const typedef struct {

```

```

2557         int __dummy;}
2558         pthread_condattr_t *, int *);
2559 extern int pthread_condattr_setpshared(pthread_condattr_t *, int);
2560 extern int pthread_mutexattr_getpshared(const typedef struct {
2561         int __mutexkind;}
2562         pthread_mutexattr_t *, int *);
2563 extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
2564 extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *, const struct
2565 timespec {
2566         time_t tv_sec; long int
2567 tv_nsec;})
2568
2569         *);
2570 extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *, const struct
2571 timespec {
2572         time_t tv_sec; long int
2573 tv_nsec;})
2574
2575         *);
2576 extern int __register_atfork(void (*prepare) (void)
2577         , void (*parent) (void)
2578         , void (*child) (void)
2579         , void *);
2580 extern int pthread_setschedprio(typedef unsigned long int pthread_t,
2581 int);

```

11.7.2 semaphore.h

```

2582
2583 extern int sem_close(sem_t *);
2584 extern int sem_destroy(sem_t *);
2585 extern int sem_getvalue(sem_t *, int *);
2586 extern int sem_init(sem_t *, int, unsigned int);
2587 extern sem_t *sem_open(const char *, int, ...);
2588 extern int sem_post(sem_t *);
2589 extern int sem_trywait(sem_t *);
2590 extern int sem_unlink(const char *);
2591 extern int sem_wait(sem_t *);
2592 extern int sem_timedwait(sem_t *, const struct timespec *);

```

11.8 Interfaces for libgcc_s

ISO POSIX (2003) Table 11-31

11.7 Interfaces for libgcc_s

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

Table 11-31 libgcc_s Definition

| | |
|----------|---------------|
| Library: | libgcc_s |
| SONAME: | libgcc_s.so.1 |

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

[LSB] ~~this specification~~ This Specification

11.78.1 Unwind Library

11.78.1.1 Interfaces for Unwind Library

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

Table 11-32 libgcc_s - Unwind Library Function Interfaces

| | | | | |
|---|--|---|--|---|
| <code>_Unwind_Backtrace(GCC_3.3)</code> [1] | <code>_Unwind_ForcedUnwind(GCC_3.0)</code> [1] | <code>_Unwind_GetGR(GCC_3.0)</code> [1] | <code>_Unwind_GetRegionStart(GCC_3.0)</code> [1] | <code>_Unwind_Resume_or_Rethrow(GCC_3.3)</code> [1] |
| <code>_Unwind_DeleteException(GCC_3.0)</code> [1] | <code>_Unwind_GetBSP(GCC_3.3.2)</code> [1] | <code>_Unwind_GetIP(GCC_3.0)</code> [1] | <code>_Unwind_RaiseException(GCC_3.0)</code> [1] | <code>_Unwind_SetGR(GCC_3.0)</code> [1] |
| <code>_Unwind_FindEnclosingFunction(GCC_3.3)</code> [1] | <code>_Unwind_GetCFA(GCC_3.3)</code> [1] | <code>_Unwind_GetLanguageSpecificData(GCC_3.0)</code> [1] | <code>_Unwind_Resume(GCC_3.0)</code> [1] | <code>_Unwind_SetIP(GCC_3.0)</code> [1] |

Referenced Specification(s)

[1]:

| | | | |
|---|---|---|--|
| <code>_Unwind_Backtrace(GCC_3.3)</code> [LSB] | <code>_Unwind_DeleteException(GCC_3.0)</code> [LSB] | <code>_Unwind_FindEnclosingFunction(GCC_3.3)</code> [LSB] | <code>_Unwind_ForcedUnwind(GCC_3.0)</code> [LSB] |
| <code>_Unwind_GetBSP(GCC_3.3.2)</code> [LSB] | <code>_Unwind_GetCFA(GCC_3.3)</code> [LSB] | <code>_Unwind_GetGR(GCC_3.0)</code> [LSB] | <code>_Unwind_GetIP(GCC_3.0)</code> [LSB] |
| <code>_Unwind_GetLanguageSpecificData(GCC_3.0)</code> [LSB] | <code>_Unwind_GetRegionStart(GCC_3.0)</code> [LSB] | <code>_Unwind_RaiseException(GCC_3.0)</code> [LSB] | <code>_Unwind_Resume(GCC_3.0)</code> [LSB] |
| <code>_Unwind_Resume_or_Rethrow(GCC_3.3)</code> [LSB] | <code>_Unwind_SetGR(GCC_3.0)</code> [LSB] | <code>_Unwind_SetIP(GCC_3.0)</code> [LSB] | |

11.9 Data Definitions for libgcc_s

This section defines global identifiers and their values that are associated with interfaces contained in libgcc_s. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an 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 ~~this specification~~ ISO C (1999)

11.8 Interface Definitions for libgcc_s

2620 The following interfaces are included in libgcc_s and are defined by this
2621 specification. Unless otherwise noted, these interfaces shall be included in the source
2622 standard.

2623 Other interfaces listed above for libgcc_s shall behave as described in the referenced
2624 base document.

11.9 Interfaces for libdl

2625 C Language as the reference programming language, and data definitions are
2626 specified in ISO C format. The C language is used here as a convenient notation.
2627 Using a C language description of these data objects does not preclude their use by
2628 other programming languages.

11.9.1 unwind.h

```
2629
2630     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2631     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2632     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2633     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2634                                             _Unwind_Stop_Fn, void *);
2635     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2636     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2637     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2638     _Unwind_Context
2639                                             *);
2640     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2641     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2642     _Unwind_Exception
2643                                             *);
2644     extern void _Unwind_Resume(struct _Unwind_Exception *);
2645     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2646     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2647     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2648     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2649     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2650                                             _Unwind_Stop_Fn, void *);
2651     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2652     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2653     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2654     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2655     _Unwind_Context
2656                                             *);
2657     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2658     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2659     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2660     _Unwind_Exception
2661                                             *);
2662     extern void _Unwind_Resume(struct _Unwind_Exception *);
2663     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2664     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2665     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2666                                             _Unwind_Stop_Fn, void *);
2667     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2668     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2669     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2670     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2671     _Unwind_Context
2672                                             *);
```

```

2673     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2674     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2675     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2676     _Unwind_Exception
2677     *);
2678     extern void _Unwind_Resume(struct _Unwind_Exception *);
2679     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2680     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2681     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2682     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2683     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2684     _Unwind_Stop_Fn, void *);
2685     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2686     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2687     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2688     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2689     _Unwind_Context
2690     *);
2691     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2692     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2693     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2694     _Unwind_Exception
2695     *);
2696     extern void _Unwind_Resume(struct _Unwind_Exception *);
2697     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2698     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2699     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2700     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2701     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2702     _Unwind_Stop_Fn, void *);
2703     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2704     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2705     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2706     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2707     _Unwind_Context
2708     *);
2709     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2710     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2711     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2712     _Unwind_Exception
2713     *);
2714     extern void _Unwind_Resume(struct _Unwind_Exception *);
2715     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2716     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2717     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2718     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2719     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2720     _Unwind_Stop_Fn, void *);
2721     extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2722     extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2723     extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2724     extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2725     extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2726     extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2727     extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2728     _Unwind_Exception
2729     *);
2730     extern void _Unwind_Resume(struct _Unwind_Exception *);
2731     extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2732     extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2733     extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2734     extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2735     extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2736     _Unwind_Stop_Fn, void *);

```

```

2737 extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2738 extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2739 extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2740 extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2741 extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2742 extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2743 extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2744     _Unwind_Exception
2745         *);
2746 extern void _Unwind_Resume(struct _Unwind_Exception *);
2747 extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2748 extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2749 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2750     *);
2751 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2752     *);
2753 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2754     *);
2755 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2756     *);
2757 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2758     *);
2759 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2760     *);
2761 extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2762     *);
2763 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2764 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2765 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2766 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2767 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2768 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2769 extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
2770 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2771     _Unwind_Exception *);
2772 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2773     _Unwind_Exception *);
2774 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2775     _Unwind_Exception *);
2776 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2777     _Unwind_Exception *);
2778 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2779     _Unwind_Exception *);
2780 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2781     _Unwind_Exception *);
2782 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2783     _Unwind_Exception *);
2784 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2785     _Unwind_Exception *);
2786 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2787     _Unwind_Exception *);
2788 extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct
2789     _Unwind_Exception *);
2790 extern void *_Unwind_FindEnclosingFunction(void *);
2791 extern void *_Unwind_FindEnclosingFunction(void *);
2792 extern void *_Unwind_FindEnclosingFunction(void *);
2793 extern void *_Unwind_FindEnclosingFunction(void *);
2794 extern void *_Unwind_FindEnclosingFunction(void *);
2795 extern void *_Unwind_FindEnclosingFunction(void *);
2796 extern void *_Unwind_FindEnclosingFunction(void *);
2797 extern void *_Unwind_FindEnclosingFunction(void *);
2798 extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);

```

11.10 Interface Definitions for `libgcc_s`

2799 The interfaces defined on the following pages are included in `libgcc_s` and are
 2800 defined by this specification. Unless otherwise noted, these interfaces shall be
 2801 included in the source standard.

2802 Other interfaces listed in ~~Table 11-33~~Section 11.8 shall behave as described in the
 2803 referenced base document.

`_Unwind_DeleteException`

Name

2804 `_Unwind_DeleteException` – private C++ error handling method

Synopsis

2805 `void _Unwind_DeleteException(struct _Unwind_Exception * object);`

Description

2806 `_Unwind_DeleteException()` deletes the given exception *object*. If a given
 2807 runtime resumes normal execution after catching a foreign exception, it will not
 2808 know how to delete that exception. Such an exception shall be deleted by calling
 2809 `_Unwind_DeleteException()`. This is a convenience function that calls the function
 2810 pointed to by the *exception_cleanup* field of the exception header.

`_Unwind_ForcedUnwind`

Name

2811 `_Unwind_ForcedUnwind` — private C++ error handling method

Synopsis

```
2812 _Unwind_Reason_Code _Unwind_ForcedUnwind(struct _Unwind_Exception *  
2813 object, _Unwind_Stop_Fn stop, void * stop_parameter);
```

Description

2814 `_Unwind_ForcedUnwind()` raises an exception for forced unwinding, passing along
2815 the given exception *object*, which should have its *exception_class* and
2816 *exception_cleanup* fields set. The exception *object* has been allocated by the
2817 language-specific runtime, and has a language-specific format, except that it shall
2818 contain an `_Unwind_Exception` struct.

2819 Forced unwinding is a single-phase process. *stop* and *stop_parameter* control the
2820 termination of the unwind process instead of the usual personality routine query.
2821 *stop* is called for each unwind frame, with the parameteres described for the usual
2822 personality routine below, plus an additional *stop_parameter*.

Return Value

2823 When *stop* identifies the destination frame, it transfers control to the user code as
2824 appropriate without returning, normally after calling `_Unwind_DeleteException()`.
2825 If not, then it should return an `_Unwind_Reason_Code` value.

2826 If *stop* returns any reason code other than `_URC_NO_REASON`, then the stack state is
2827 indeterminate from the point of view of the caller of `_Unwind_ForcedUnwind()`.
2828 Rather than attempt to return, therefore, the unwind library should use the
2829 *exception_cleanup* entry in the exception, and then call `abort()`.

`_URC_NO_REASON`

2830
2831 This is not the destination from. The unwind runtime will call frame's
2832 personality routine with the `_UA_FORCE_UNWIND` and `_UA_CLEANUP_PHASE` flag
2833 set in *actions*, and then unwind to the next frame and call the `stop()` function
2834 again.

`_URC_END_OF_STACK`

2835
2836 In order to allow `_Unwind_ForcedUnwind()` to perform special processing
2837 when it reaches the end of the stack, the unwind runtime will call it after the last
2838 frame is rejected, with a `NULL` stack pointer in the context, and the `stop()`
2839 function shall catch this condition. It may return this code if it cannot handle
2840 end-of-stack.

`_URC_FATAL_PHASE2_ERROR`

2841
2842 The `stop()` function may return this code for other fatal conditions like stack
2843 corruption.

_Unwind_GetGR

Name

2844 `_Unwind_GetGR` – private C++ error handling method

Synopsis

2845 `_Unwind_Word _Unwind_GetGR(struct _Unwind_Context * context, int index);`

Description

2846 `_Unwind_GetGR()` returns data at *index* found in *context*. The register is identified
2847 by its index: 0 to 31 are for the fixed registers, and 32 to 127 are for the stacked
2848 registers.

2849 During the two phases of unwinding, only GR1 has a guaranteed value, which is the
2850 global pointer of the frame referenced by the unwind *context*. If the register has its
2851 NAT bit set, the behavior is unspecified.

_Unwind_GetIP

Name

2852 `_Unwind_GetIP` – private C++ error handling method

Synopsis

2853 `_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);`

Description

2854 `_Unwind_GetIP()` returns the instruction pointer value for the routine identified by
2855 the unwind *context*.

_Unwind_GetLanguageSpecificData

Name

2856 `_Unwind_GetLanguageSpecificData` – private C++ error handling method

Synopsis

2857 `_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *`
2858 `context, uint value);`

Description

2859 `_Unwind_GetLanguageSpecificData()` returns the address of the language specific
2860 data area for the current stack frame.

_Unwind_GetRegionStart

Name

2861 `_Unwind_GetRegionStart` – private C++ error handling method

Synopsis

2862 `_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);`

Description

2863 `_Unwind_GetRegionStart()` routine returns the address (i.e., 0) of the beginning of
2864 the procedure or code fragment described by the current unwind descriptor block.

_Unwind_RaiseException

Name

2865 `_Unwind_RaiseException` – private C++ error handling method

Synopsis

2866 `_Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *
2867 object);`

Description

2868 `_Unwind_RaiseException()` raises an exception, passing along the given exception
2869 *object*, which should have its *exception_class* and *exception_cleanup* fields set.
2870 The exception object has been allocated by the language-specific runtime, and has a
2871 language-specific format, exception that it shall contain an `_Unwind_Exception`.

Return Value

2872 `_Unwind_RaiseException()` does not return unless an error condition is found. If
2873 an error condition occurs, an `_Unwind_Reason_Code` is returned:

`_URC_END_OF_STACK`

2875 The unwinder encountered the end of the stack during phase one without
2876 finding a handler. The unwind runtime will not have modified the stack. The
2877 C++ runtime will normally call `uncaught_exception()` in this case.

`_URC_FATAL_PHASE1_ERROR`

2879 The unwinder encountered an unexpected error during phase one, because of
2880 something like stack corruption. The unwind runtime will not have modified
2881 the stack. The C++ runtime will normally call `terminate()` in this case.

`_URC_FATAL_PHASE2_ERROR`

2883 The unwinder encountered an unexpected error during phase two. This is
2884 usually a *throw*, which will call `terminate()`.

_Unwind_Resume

Name

2885 `_Unwind_Resume` – private C++ error handling method

Synopsis

2886 `void _Unwind_Resume(struct _Unwind_Exception * object);`

Description

2887 `_Unwind_Resume()` resumes propagation of an existing exception *object*. A call to
2888 this routine is inserted as the end of a landing pad that performs cleanup, but does
2889 not resume normal execution. It causes unwinding to proceed further.

_Unwind_SetGR

Name

2890 `_Unwind_SetGR` – private C++ error handling method

Synopsis

2891 `void _Unwind_SetGR(struct _Unwind_Context * context, int index, uint value);`

Description

2892 `_Unwind_SetGR()` sets the *value* of the register *indexed* for the routine identified by
2893 the unwind *context*.

_Unwind_SetIP

Name

2894 `_Unwind_SetIP` – private C++ error handling method

Synopsis

2895 `void _Unwind_SetIP(struct _Unwind_Context * context, uint value);`

Description

2896 `_Unwind_SetIP()` sets the *value* of the instruction pointer for the routine identified
2897 by the unwind *context*.

11.11 Interfaces for libdl

2898 **Table 11-33** defines the library name and shared object name for the libdl library

2899 **Table 11-33 libdl Definition**

| | |
|----------|------------|
| Library: | libdl |
| SONAME: | libdl.so.2 |

2901 The behavior of the interfaces in this library is specified by the following specifica-
2902 tions:

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

11.9-11.1 Dynamic Loader

11.911.1.1 Interfaces for Dynamic Loader

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

Table 11-34 libdl - Dynamic Loader Function Interfaces

| | | | | |
|-----------------------------------|------------------------------------|------------------------------------|-----------------------------------|----------------------------------|
| dladdr(GLIB C_2.0) [1] | dlclose(GLIB C_2.0) [2] | dLError(GLIB C_2.0) [2] | dlopen(GLIB C_2.1) [1] | dlsym(GLIBC _2.0) [1] |
|-----------------------------------|------------------------------------|------------------------------------|-----------------------------------|----------------------------------|

~~Referenced Specification(s)~~

~~[1]~~

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

11.12 Data Definitions for libdl

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.

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 ~~this specification~~ ISO C (1999)

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

11.12.1 dlfcn.h

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

11.13 Interfaces for libcrypt

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

11.10 Interfaces for libcrypt

2935 **Table 11-35** defines the library name and shared object name for the libcrypt library

2936 **Table 11-35 libcrypt Definition**

| | |
|----------|---------------|
| Library: | libcrypt |
| SONAME: | libcrypt.so.1 |

2937

2938 The behavior of the interfaces in this library is specified by the following specifica-
2939 tions:

2940 **[SUSv3]** ISO POSIX (2003)

11.4013.1 Encryption

11.4013.1.1 Interfaces for Encryption

2941

2942 An LSB conforming implementation shall provide the architecture specific functions
2943 for Encryption specified in Table 11-36, with the full mandatory functionality as
2944 described in the referenced underlying specification.

Table 11-36 libcrypt - Encryption Function Interfaces

2945

| | | | | |
|---------------------------------|-----------------------------------|----------------------------------|--|--|
| crypt(GLIBC_2.0) [1] | encrypt(GLIBC_2.0) [1] | setkey(GLIBC_2.0) [1] | | |
|---------------------------------|-----------------------------------|----------------------------------|--|--|

2946

Referenced Specification(s)

2947

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

2948

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

2949

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_stream *, int);
```



```

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

```

12.3 Interfaces for libncurses

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

72 **Table 12-2 libncurses Definition**

| | |
|-------------|-----------------|
| 73 Library: | libncurses |
| SONAME: | libncurses.so.5 |

12.23.1 Curses

74 12.23.1.1 Interfaces for Curses

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

12.34 Data Definitions for libncurses

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

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

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

12.4.1 curses.h

```

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

```

```

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

```

```

206     extern int mvwaddchstr(WINDOW *, int, int, const chtype *);
207     extern int mvwaddnstr(WINDOW *, int, int, const char *, int);
208     extern int mvwaddstr(WINDOW *, int, int, const char *);
209     extern int mvwchgat(WINDOW *, int, int, int, attr_t, short, const void
210     *);
211     extern int mvwdelch(WINDOW *, int, int);
212     extern int mvwgetch(WINDOW *, int, int);
213     extern int mvwgetnstr(WINDOW *, int, int, char *, int);
214     extern int mvwgetstr(WINDOW *, int, int, char *);
215     extern int mvwhline(WINDOW *, int, int, chtype, int);
216     extern int mvwin(WINDOW *, int, int);
217     extern chtype mvwinch(WINDOW *, int, int);
218     extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
219     extern int mvwinchstr(WINDOW *, int, int, chtype *);
220     extern int mvwinnstr(WINDOW *, int, int, char *, int);
221     extern int mvwinsch(WINDOW *, int, int, chtype);
222     extern int mvwinsnstr(WINDOW *, int, int, const char *, int);
223     extern int mvwinsstr(WINDOW *, int, int, const char *);
224     extern int mvwinstr(WINDOW *, int, int, char *);
225     extern int mvwprintw(WINDOW *, int, int, char *, ...);
226     extern int mvwscanw(WINDOW *, int, int, const char *, ...);
227     extern int mvwvline(WINDOW *, int, int, chtype, int);
228     extern int napms(int);
229     extern WINDOW *newpad(int, int);
230     extern SCREEN *newterm(const char *, FILE *, FILE *);
231     extern WINDOW *newwin(int, int, int, int);
232     extern int nl(void);
233     extern int nocbreak(void);
234     extern int nodelay(WINDOW *, bool);
235     extern int noecho(void);
236     extern int nonl(void);
237     extern void noqiflush(void);
238     extern int noraw(void);
239     extern int notimeout(WINDOW *, bool);
240     extern int overlay(const WINDOW *, WINDOW *);
241     extern int overwrite(const WINDOW *, WINDOW *);
242     extern int pair_content(short, short *, short *);
243     extern int pechochar(WINDOW *, chtype);
244     extern int pnoutrefresh(WINDOW *, int, int, int, int, int, int);
245     extern int prefresh(WINDOW *, int, int, int, int, int, int);
246     extern int printw(char *, ...);
247     extern int putwin(WINDOW *, FILE *);
248     extern void qiflush(void);
249     extern int raw(void);
250     extern int redrawwin(WINDOW *);
251     extern int refresh(void);
252     extern int resetty(void);
253     extern int reset_prog_mode(void);
254     extern int reset_shell_mode(void);
255     extern int ripoffline(int, int (*init) (WINDOW *, int)
256     );
257     extern int savetty(void);
258     extern int scanw(const char *, ...);
259     extern int scr_dump(const char *);
260     extern int scr_init(const char *);
261     extern int scr1(int);
262     extern int scroll(WINDOW *);
263     extern int scrollok(WINDOW *, typedef unsigned char bool);
264     extern int scr_restore(const char *);
265     extern int scr_set(const char *);
266     extern int setscreg(int, int);
267     extern SCREEN *set_term(SCREEN *);
268     extern int slk_attroff(const typedef unsigned long int chtype);
269     extern int slk_attron(const typedef unsigned long int chtype);

```

```

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

```

```

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

```

12.4.2 term.h

```

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

```

395 `extern TERMINAL *cur_term(void);`

12.5 Interfaces for libutil

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

397 **Table 12-3 libutil Definition**

| | |
|----------|--------------|
| Library: | libutil |
| SONAME: | libutil.so.1 |

398

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

401 ~~[LSB] this specification~~ This Specification

12.35.1 Utility Functions

12.35.1.1 Interfaces for Utility Functions

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

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

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

407

408 *Referenced Specification(s)*

409 ~~[1], this specification~~

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

410

V Package Format and Installation

13 Software Installation

13.1 Package Dependencies

1 The LSB runtime environment shall provide the following dependencies.

2 `lsb-core-ia64`

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-ia64`.

13.2 Package Architecture Considerations

8 All packages must specify an architecture of `IA64`. A LSB runtime environment must
9 accept an architecture of `ia64` even if the native architecture is different.

10 The `archnum` value in the Lead Section shall be `0x0009`.

Annex A Alphabetical Listing of Interfaces

A.1 libgcc_s

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

2 ~~this specification~~ This Specification [LSB]

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

| | | |
|---|--|---|
| _Unwind_Backtrace_Unwind_Backtrace [1]LSB] | _Unwind_GetCFA_Unwind_GetCFA [1]LSB] | _Unwind_RaiseException [1]LSB] |
| _Unwind_DeleteException [1]LSB] | _Unwind_GetGR_Unwind_GetGR [1]LSB] | _Unwind_Resume_Unwind_Resume [1]LSB] |
| _Unwind_FindEnclosingFunction [1]LSB] | _Unwind_GetIP_Unwind_GetIP [1]LSB] | _Unwind_Resume_or_Rethrow [1]LSB] |
| _Unwind_ForcedUnwind_Unwind_ForcedUnwind [1]LSB] | _Unwind_GetLanguageSpecificData [1]LSB] | _Unwind_SetGR_Unwind_SetGR [1]LSB] |
| _Unwind_GetBSP_Unwind_GetBSP [1]LSB] | _Unwind_GetRegionStart [1]LSB] | _Unwind_SetIP_Unwind_SetIP [1]LSB] |

A.2 libm

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

6 ISO C (1999) [ISO C99]

7 ISO POSIX (2003) [SUSv3]

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

| | | |
|---|---|-------------------------------|
| __fpclassify__fpclassify [1]ISO C99] | __signbit__signbit [1]ISO C99] | exp2exp2 [1]SUSv3] |
|---|---|-------------------------------|

Annex B GNU Free Documentation License (Informative)

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

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

B.1 PREAMBLE

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

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

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

B.2 APPLICABILITY AND DEFINITIONS

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

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

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

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

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

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

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

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

B.3 VERBATIM COPYING

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

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

B.4 COPYING IN QUANTITY

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

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

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

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

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

B.5 MODIFICATIONS

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

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

- 134 given in the Document for previous versions it was based on. These may be
135 placed in the "History" section. You may omit a network location for a work
136 that was published at least four years before the Document itself, or if the
137 original publisher of the version it refers to gives permission.
- 138 K. In any section entitled "Acknowledgements" or "Dedications", preserve the
139 section's title, and preserve in the section all the substance and tone of each of
140 the contributor acknowledgements and/or dedications given therein.
- 141 L. Preserve all the Invariant Sections of the Document, unaltered in their text and
142 in their titles. Section numbers or the equivalent are not considered part of the
143 section titles.
- 144 M. Delete any section entitled "Endorsements". Such a section may not be
145 included in the Modified Version.
- 146 N. Do not retitle any existing section as "Endorsements" or to conflict in title with
147 any Invariant Section.
- 148 If the Modified Version includes new front-matter sections or appendices that
149 qualify as Secondary Sections and contain no material copied from the Document,
150 you may at your option designate some or all of these sections as invariant. To do
151 this, add their titles to the list of Invariant Sections in the Modified Version's license
152 notice. These titles must be distinct from any other section titles.
- 153 You may add a section entitled "Endorsements", provided it contains nothing but
154 endorsements of your Modified Version by various parties—for example, statements
155 of peer review or that the text has been approved by an organization as the
156 authoritative definition of a standard.
- 157 You may add a passage of up to five words as a Front-Cover Text, and a passage of
158 up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the
159 Modified Version. Only one passage of Front-Cover Text and one of Back-Cover
160 Text may be added by (or through arrangements made by) any one entity. If the
161 Document already includes a cover text for the same cover, previously added by you
162 or by arrangement made by the same entity you are acting on behalf of, you may not
163 add another; but you may replace the old one, on explicit permission from the
164 previous publisher that added the old one.
- 165 The author(s) and publisher(s) of the Document do not by this License give
166 permission to use their names for publicity for or to assert or imply endorsement of
167 any Modified Version.

B.6 COMBINING DOCUMENTS

- 168 You may combine the Document with other documents released under this License,
169 under the terms defined in section 4 above for modified versions, provided that you
170 include in the combination all of the Invariant Sections of all of the original
171 documents, unmodified, and list them all as Invariant Sections of your combined
172 work in its license notice.
- 173 The combined work need only contain one copy of this License, and multiple
174 identical Invariant Sections may be replaced with a single copy. If there are multiple
175 Invariant Sections with the same name but different contents, make the title of each
176 such section unique by adding at the end of it, in parentheses, the name of the
177 original author or publisher of that section if known, or else a unique number. Make
178 the same adjustment to the section titles in the list of Invariant Sections in the license
179 notice of the combined work.

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

B.7 COLLECTIONS OF DOCUMENTS

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

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

B.8 AGGREGATION WITH INDEPENDENT WORKS

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

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

B.9 TRANSLATION

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

B.10 TERMINATION

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

B.11 FUTURE REVISIONS OF THIS LICENSE

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

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

B.12 How to use this License for your documents

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

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

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

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