Linux Standard Base Core Specification
for IA32 3.1
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>vi</td>
</tr>
<tr>
<td>Introduction</td>
<td>vii</td>
</tr>
<tr>
<td><strong>I Introductory Elements</strong></td>
<td>8</td>
</tr>
<tr>
<td>1 Scope</td>
<td></td>
</tr>
<tr>
<td>1.1 General</td>
<td>9</td>
</tr>
<tr>
<td>1.2 Module Specific Scope</td>
<td>9</td>
</tr>
<tr>
<td>2 References</td>
<td></td>
</tr>
<tr>
<td>2.1 Normative References</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Informative References/Bibliography</td>
<td>12</td>
</tr>
<tr>
<td>3 Requirements</td>
<td></td>
</tr>
<tr>
<td>3.1 Relevant Libraries</td>
<td>15</td>
</tr>
<tr>
<td>3.2 LSB Implementation Conformance</td>
<td>15</td>
</tr>
<tr>
<td>3.3 LSB Application Conformance</td>
<td>16</td>
</tr>
<tr>
<td>4 Definitions</td>
<td></td>
</tr>
<tr>
<td>5 Terminology</td>
<td>19</td>
</tr>
<tr>
<td>6 Documentation Conventions</td>
<td>21</td>
</tr>
<tr>
<td><strong>II Executable and Linking Format (ELF)</strong></td>
<td>22</td>
</tr>
<tr>
<td>7 Introduction</td>
<td>23</td>
</tr>
<tr>
<td>8 Low Level System Information</td>
<td>24</td>
</tr>
<tr>
<td>8.1 Machine Interface</td>
<td>24</td>
</tr>
<tr>
<td>8.2 Function Calling Sequence</td>
<td>25</td>
</tr>
<tr>
<td>8.3 Operating System Interface</td>
<td>26</td>
</tr>
<tr>
<td>8.4 Process Initialization</td>
<td>27</td>
</tr>
<tr>
<td>8.5 Coding Examples</td>
<td>28</td>
</tr>
<tr>
<td>8.6 C Stack Frame</td>
<td>29</td>
</tr>
<tr>
<td>8.7 Debug Information</td>
<td>29</td>
</tr>
<tr>
<td>9 Object Format</td>
<td></td>
</tr>
<tr>
<td>9.1 Introduction</td>
<td>30</td>
</tr>
<tr>
<td>9.2 ELF Header</td>
<td>30</td>
</tr>
<tr>
<td>9.3 Special Sections</td>
<td>30</td>
</tr>
<tr>
<td>9.4 Symbol Table</td>
<td>31</td>
</tr>
<tr>
<td>9.5 Relocation</td>
<td>31</td>
</tr>
<tr>
<td>10 Program Loading and Dynamic Linking</td>
<td></td>
</tr>
<tr>
<td>10.1 Introduction</td>
<td>32</td>
</tr>
<tr>
<td>10.2 Program Header</td>
<td>32</td>
</tr>
<tr>
<td>10.3 Program Loading</td>
<td>32</td>
</tr>
<tr>
<td>10.4 Dynamic Linking</td>
<td>32</td>
</tr>
<tr>
<td><strong>III Base Libraries</strong></td>
<td>34</td>
</tr>
<tr>
<td>11 Libraries</td>
<td></td>
</tr>
<tr>
<td>11.1 Program Interpreter/Dynamic Linker</td>
<td>35</td>
</tr>
<tr>
<td>11.2 Interfaces for libc</td>
<td>35</td>
</tr>
<tr>
<td>11.3 Data Definitions for libc</td>
<td>49</td>
</tr>
<tr>
<td>11.4 Interfaces for libm</td>
<td>75</td>
</tr>
<tr>
<td>11.5 Data Definitions for libm</td>
<td>79</td>
</tr>
<tr>
<td>11.6 Interface Definitions for libm</td>
<td>86</td>
</tr>
<tr>
<td>11.7 Interfaces for libpthread</td>
<td>86</td>
</tr>
<tr>
<td>11.8 Data Definitions for libpthread</td>
<td>89</td>
</tr>
<tr>
<td>11.9 Interfaces for libgcc_s</td>
<td>93</td>
</tr>
<tr>
<td>11.10 Data Definitions for libgcc_s</td>
<td>94</td>
</tr>
</tbody>
</table>
11.11 Interface Definitions for libgcc_s .......................................................... 97
11.12 Interfaces for libdl ............................................................................... 102
11.13 Data Definitions for libdl ..................................................................... 103
11.14 Interfaces for libcrypto ....................................................................... 103

IV Utility Libraries ...................................................................................... 104

12 Libraries ................................................................................................. 105
12.1 Interfaces for libz ............................................................................... 105
12.2 Data Definitions for libz .................................................................... 105
12.3 Interfaces for libncurses .................................................................... 106
12.4 Data Definitions for libncurses ............................................................ 106
12.5 Interfaces for libutil .......................................................................... 112

V Package Format and Installation .............................................................. 113

13 Software Installation ............................................................................... 114
13.1 Package Dependencies ....................................................................... 114
13.2 Package Architecture Considerations ................................................. 114

A Alphabetical Listing of Interfaces ........................................................... 115
A.1 libgcc_s .............................................................................................. 115
A.2 libm ........................................................................................................ 115

B GNU Free Documentation License (Informative) ...................................... 116
B.1 PREAMBLE ......................................................................................... 116
B.2 APPLICABILITY AND DEFINITIONS ................................................ 116
B.3 VERBATIM COPYING ......................................................................... 117
B.4 COPYING IN QUANTITY .................................................................... 117
B.5 MODIFICATIONS ................................................................................. 118
B.6 COMBINING DOCUMENTS .................................................................... 119
B.7 COLLECTIONS OF DOCUMENTS ...................................................... 120
B.8 AGGREGATION WITH INDEPENDENT WORKS .................................. 120
B.9 TRANSLATION .................................................................................... 120
B.10 TERMINATION .................................................................................... 120
B.11 FUTURE REVISIONS OF THIS LICENSE ......................................... 121
B.12 How to use this License for your documents ....................................... 121
## List of Tables

- **2-1 Normative References** ........................................................................................................... 10
- **2-2 Other References** .................................................................................................................... 12
- **3-1 Standard Library Names** ......................................................................................................... 15
- **8-1 Scalar Types** .......................................................................................................................... 24
- **9-1 ELF Special Sections** .............................................................................................................. 30
- **9-2 Additional Special Sections** ................................................................................................... 31
- **11-1 libc Definition** ...................................................................................................................... 35
- **11-2 libc - RPC Function Interfaces** ............................................................................................ 35
- **11-3 libc - System Calls Function Interfaces** ............................................................................... 36
- **11-4 libc - Standard I/O Function Interfaces** ............................................................................... 38
- **11-5 libc - Standard I/O Data Interfaces** ...................................................................................... 39
- **11-6 libc - Signal Handling Function Interfaces** .......................................................................... 39
- **11-7 libc - Signal Handling Data Interfaces** ............................................................................... 40
- **11-8 libc - Localization Functions Function Interfaces** .............................................................. 40
- **11-9 libc - Localization Functions Data Interfaces** .................................................................... 41
- **11-10 libc - Socket Interface Function Interfaces** ....................................................................... 41
- **11-11 libc - Wide Characters Function Interfaces** ...................................................................... 41
- **11-12 libc - String Functions Function Interfaces** ..................................................................... 43
- **11-13 libc - IPC Functions Function Interfaces** ......................................................................... 44
- **11-14 libc - Regular Expressions Function Interfaces** ............................................................... 44
- **11-15 libc - Character Type Functions Function Interfaces** ...................................................... 44
- **11-16 libc - Time Manipulation Function Interfaces** .................................................................. 45
- **11-17 libc - Time Manipulation Data Interfaces** ........................................................................ 45
- **11-18 libc - Terminal Interface Functions Function Interfaces** .............................................. 45
- **11-19 libc - System Database Interface Function Interfaces** ................................................. 46
- **11-20 libc - Language Support Function Interfaces** ................................................................... 47
- **11-21 libc - Large File Support Function Interfaces** .................................................................. 47
- **11-22 libc - Standard Library Function Interfaces** ..................................................................... 47
- **11-23 libc - Standard Library Data Interfaces** ........................................................................... 49
- **11-24 libm Definition** .................................................................................................................. 75
- **11-25 libm - Math Function Interfaces** ....................................................................................... 76
- **11-26 libm - Math Data Interfaces** ............................................................................................. 79
- **11-27 libpthread Definition** ........................................................................................................ 86
- **11-28 libpthread - Realtime Threads Function Interfaces** ........................................................ 86
- **11-29 libpthread - Posix Threads Function Interfaces** ............................................................. 87
- **11-30 libpthread - Thread aware versions of libc interfaces Function Interfaces** .................. 88
- **11-31 libgcc_s Definition** .......................................................................................................... 93
- **11-32 libgcc_s - Unwind Library Function Interfaces** ................................................................ 93
- **11-33 libdl Definition** ................................................................................................................. 102
- **11-34 libdl - Dynamic Loader Function Interfaces** .................................................................... 102
- **11-35 libcrypt Definition** ............................................................................................................ 103
- **11-36 libcrypt - Encryption Function Interfaces** ...................................................................... 103
- **12-1 libz Definition** .................................................................................................................... 105
- **12-2 libncurses Definition** ........................................................................................................... 106
- **12-3 libutil Definition** ................................................................................................................ 112
- **12-4 libutil - Utility Functions Function Interfaces** .................................................................. 112
- **A-1 libgcc_s Function Interfaces** .............................................................................................. 115
- **A-2 libm Function Interfaces** ....................................................................................................... 115
Foreword

This is version 3.1 of the Linux Standard Base Core Specification for IA32. This specification is part of a family of specifications under the general title "Linux Standard Base". Developers of applications or implementations interested in using the LSB trademark should see the Free Standards Group Certification Policy for details.
Introduction

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

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

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

- The first number (x) is the major version number. All versions with the same major version number should share binary compatibility. Any addition or deletion of a new library results in a new version number. Interfaces marked as deprecated may be removed from the specification at a major version change.

- The second number (y) is the minor version number. Individual interfaces may be added if all certified implementations already had that (previously undocumented) interface. Interfaces may be marked as deprecated at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.

- The third number (z), if present, is the editorial level. Only editorial changes should be included in such versions.

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

Implementors are strongly encouraged to make use of symbol versioning to permit simultaneous support of applications conforming to different releases of this specification.
I Introductory Elements
1 Scope

1.1 General

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

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

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

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

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

1.2 Module Specific Scope

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filesystem Hierarchy Standard</td>
<td>Filesystem Hierarchy Standard (FHS) 2.3</td>
<td><a href="http://www.pathname.com/fhs/">http://www.pathname.com/fhs/</a></td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td>URL</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>Itanium C++ ABI</td>
<td>Itanium C++ ABI (Revision 1.83)</td>
<td><a href="http://refs">http://refs</a> specs.freestand ards.org/cxxabi-1.83.htm 1</td>
</tr>
<tr>
<td>SVID Issue 4</td>
<td>System V Interface</td>
<td></td>
</tr>
</tbody>
</table>
2 References

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
</table>

2.2 Informative References/Bibliography

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

Table 2-2 Other References

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWARF Debugging Information Format, Revision 2.0.0</td>
<td>DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)</td>
<td><a href="http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf">http://refspecs.freestandards.org/dwarf/dwarf-2.0.0.pdf</a></td>
</tr>
<tr>
<td>DWARF Debugging Information Format, Revision 3.0.0 (Draft)</td>
<td>DWARF Debugging Information Format, Revision 3.0.0 (Draft)</td>
<td><a href="http://refspecs.freestandards.org/dwarf/">http://refspecs.freestandards.org/dwarf/</a></td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td>URL</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Linux Allocated Device</td>
<td>LINUX ALLOCATED DEVICES</td>
<td><a href="http://www.lanana.org/docs/device-list/device.s.txt">http://www.lanana.org/docs/device-list/device.s.txt</a></td>
</tr>
<tr>
<td>Registry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAM</td>
<td>Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar &amp; R.Schemers (SunSoft)</td>
<td><a href="http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt">http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt</a></td>
</tr>
<tr>
<td>RFC 1831/1832 RPC &amp; XDR</td>
<td>IETF RFC 1831 &amp; 1832</td>
<td><a href="http://www.ietf.org/">http://www.ietf.org/</a></td>
</tr>
</tbody>
</table>
2 References

<table>
<thead>
<tr>
<th>Name</th>
<th>Title</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ib/</td>
</tr>
</tbody>
</table>
3 Requirements

3.1 Relevant Libraries

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

Table 3-1 Standard Library Names

<table>
<thead>
<tr>
<th>Library</th>
<th>Runtime Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>libm</td>
<td>libm.so.6</td>
</tr>
<tr>
<td>libdl</td>
<td>libdl.so.2</td>
</tr>
<tr>
<td>libcrypt</td>
<td>libcrypt.so.1</td>
</tr>
<tr>
<td>libz</td>
<td>libz.so.1</td>
</tr>
<tr>
<td>libncurses</td>
<td>libncurses.so.5</td>
</tr>
<tr>
<td>libutil</td>
<td>libutil.so.1</td>
</tr>
<tr>
<td>libc</td>
<td>libc.so.6</td>
</tr>
<tr>
<td>libpthread</td>
<td>libpthread.so.0</td>
</tr>
<tr>
<td>proginterp</td>
<td>/lib/ld-1d-so.0</td>
</tr>
<tr>
<td>libgcc_s</td>
<td>libgcc_s.so.1</td>
</tr>
</tbody>
</table>

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

3.2 LSB Implementation Conformance

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

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

A conforming implementation shall satisfy the following requirements:

• A processor architecture represents a family of related processors which may not have identical feature sets. The architecture specific supplement to this specification for a given target processor architecture describes a minimum acceptable processor. The implementation shall provide all features of this processor, whether in hardware or through emulation transparent to the application.

• The implementation shall be capable of executing compiled applications having the format and using the system interfaces described in this document.

• The implementation shall provide libraries containing the interfaces specified by this document, and shall provide a dynamic linking mechanism that allows these
3 Requirements

The interfaces to be attached to applications at runtime. All the interfaces shall behave as specified in this document.

• The map of virtual memory provided by the implementation shall conform to the requirements of this document.

• The implementation's low-level behavior with respect to function call linkage, system traps, signals, and other such activities shall conform to the formats described in this document.

• The implementation shall provide all of the mandatory interfaces in their entirety.

• The implementation may provide one or more of the optional interfaces. Each optional interface that is provided shall be provided in its entirety. The product documentation shall state which optional interfaces are provided.

• The implementation shall provide all files and utilities specified as part of this document in the format defined here and in other referenced documents. All commands and utilities shall behave as required by this document. The implementation shall also provide all mandatory components of an application's runtime environment that are included or referenced in this document.

• The implementation, when provided with standard data formats and values at a named interface, shall provide the behavior defined for those values and data formats at that interface. However, a conforming implementation may consist of components which are separately packaged and/or sold. For example, a vendor of a conforming implementation might sell the hardware, operating system, and windowing system as separately packaged items.

• The implementation may provide additional interfaces with different names. It may also provide additional behavior corresponding to data values outside the standard ranges, for standard named interfaces.

3.3 LSB Application Conformance

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

A conforming application shall satisfy the following requirements:

• Its executable files shall be either shell scripts or object files in the format defined for the Object File Format system interface.

• Its object files shall participate in dynamic linking as defined in the Program Loading and Linking System interface.

• It shall employ only the instructions, traps, and other low-level facilities defined in the Low-Level System interface as being for use by applications.

• If it requires any optional interface defined in this document in order to be installed or to execute successfully, the requirement for that optional interface shall be stated in the application's documentation.

• It shall not use any interface or data format that is not required to be provided by a conforming implementation, unless:

  • If such an interface or data format is supplied by another application through direct invocation of that application during execution, that application shall be in turn an LSB conforming application.
3 Requirements

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

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

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

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

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

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

may
is permitted; is allowed; is permissible

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

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

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

should
it is recommended that; ought to

should not
it is not recommended that; ought not to
5 Terminology

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

archLSB

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

Binary Standard

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

gLSB

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

implementation-defined

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

Shell Script

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

Source Standard

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

undefined

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

unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.
Other terms and definitions used in this document shall have the same meaning as defined in Chapter 3 of the Base Definitions volume of ISO POSIX (2003).
6 Documentation Conventions

Throughout this document, the following typographic conventions are used:

function()  
the name of a function

command  
the name of a command or utility

CONSTANT  
a constant value

parameter  
a parameter

variable  
a variable

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

ame  
the name of the interface

(symver)  
An optional symbol version identifier, if required.

[refno]  
A reference number indexing the table of referenced specifications that follows this table.

For example,

forkpty(GLIBC_2.0) [SUSv3]

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

Note: Symbol versions are defined in the architecture specific supplements only.
II Executable and Linking Format (ELF)
7 Introduction

Executable and Linking Format (ELF) defines the object format for compiled applications. This specification supplements the information found in System V ABI Update and System V ABI, IA32 Supplement, and is 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 IA32 Architecture is specified by the following documents:

- Intel® Architecture Software Developer's Manual Volume 1
- Intel® Architecture Software Developer's Manual Volume 3

Only the features of the Intel486 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 a conforming application shall not use it.

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

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

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

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

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

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

8.1.2 Data Representation

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

8.1.2.1 Byte Ordering

LSB-conforming systems and applications shall use the bit and byte ordering rules specified in Section 1.3.1 of the Intel® Architecture Software Developer's Manual Volume 1.

8.1.2.2 Fundamental Types

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

Table 8-1 Scalar Types

<table>
<thead>
<tr>
<th>Type</th>
<th>C</th>
<th>sizeof (bytes)</th>
<th>Alignment (bytes)</th>
<th>Intel386 Architecture</th>
</tr>
</thead>
</table>

...
8 Low Level System Information

<table>
<thead>
<tr>
<th>Type</th>
<th>C</th>
<th>sizeof</th>
<th>Alignment (bytes)</th>
<th>Intel386 Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integral</td>
<td>long long</td>
<td>8</td>
<td>4</td>
<td>signed double word</td>
</tr>
<tr>
<td></td>
<td>signed long long</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unsigned long long</td>
<td>8</td>
<td>4</td>
<td>unsigned double word</td>
</tr>
</tbody>
</table>

8.1.2.3 Aggregates and Unions
LSB-conforming implementations shall support aggregates and unions with alignment and padding as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.1.2.4 Bit Fields
LSB-conforming implementations shall support structure and union definitions that include bit-fields as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2 Function Calling Sequence
LSB-conforming applications shall use the function calling sequence as defined in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.1 Registers
LSB-conforming applications shall use the general registers provided by the architecture in the manner described in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.2 Floating Point Registers
LSB-conforming applications shall use the floating point registers provided by the architecture in the manner described in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.3 Stack Frame
LSB-conforming applications shall use the stack frame in the manner specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4 Arguments
8.2.4.1 Integral/Pointer
Integral and pointer arguments to functions shall be passed as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4.2 Floating Point
Floating point arguments to functions shall be passed as specified in Chapter 3 of the System V ABI, IA32 Supplement.
8.2.4.3 Struct and Union Arguments
Structure and union arguments to functions shall be passed as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.4.4 Variable Arguments
As described in Chapter 3 of the System V ABI, IA32 Supplement, LSB-conforming applications using variable argument lists shall use the facilities defined in the header file `<stdarg.h>` to deal with variable argument lists.

*Note:* This is a requirement of ISO C (1999) and ISO POSIX (2003) as well as System V ABI, IA32 Supplement.

8.2.5 Return Values

8.2.5.1 Void
As described in chapter 3 of System V ABI, IA32 Supplement, functions returning no value need not set any register to any particular value.

8.2.5.2 Integral/Pointer
Functions return scalar values (integer or pointer), shall do so as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.5.3 Floating Point
Functions return floating point values shall do so as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.2.5.4 Struct and Union
Functions that return a structure or union shall do so as specified in Chapter 3 of the System V ABI, IA32 Supplement.

8.3 Operating System Interface
LSB-conforming applications shall use the following aspects of the Operating System Interfaces as defined in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1 Virtual Address Space
LSB-conforming implementations shall support the virtual address space described in Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1.1 Page Size
LSB-conforming applications should call `sysconf()` to determine the current page size. See also Chapter 3 of the System V ABI, IA32 Supplement.

8.3.1.2 Virtual Address Assignments
LSB-conforming systems shall provide the virtual address space configuration as described in Chapter 3 of the System V ABI, IA32 Supplement (Virtual Address Assignments).

8.3.1.3 Managing the Process Stack
LSB-conforming systems shall manage the process stack as specified in Chapter 3 of the System V ABI, IA32 Supplement.
8.3.1.4 Coding Guidelines
LSB-conforming applications should follow the coding guidelines provided in
Chapter 3 of the System V ABI, IA32 Supplement.

8.3.2 Processor Execution Mode
LSB-conforming applications shall run in the user-mode ring as described in
Chapter 3 of the System V ABI, IA32 Supplement.

8.3.3 Exception Interface

8.3.3.1 Introduction
LSB-conforming system shall provide the exception interface described in Chapter 3
of the System V ABI, IA32 Supplement.

8.3.3.2 Hardware Exception Types
LSB-conforming systems shall map hardware exceptions to signals as described in
Chapter 3 of the System V ABI, IA32 Supplement.

8.3.3.3 Software Trap Types
Software generated traps are subject to the limitations described in Chapter 3 of the
System V ABI, IA32 Supplement.

8.3.4 Signal Delivery
There are no architecture specific requirements for signal delivery.

8.3.4.1 Signal Handler Interface
There are no architecture specific requirements for the signal handler interface.

8.4 Process Initialization
An LSB-conforming implementation shall cause an application to be initialized as
described in the Process Initialization section of Chapter 3 of the System V ABI, IA32
Supplement, and as described below.

8.4.1 Special Registers
The special registers shall be initialized as described in Chapter 3 of the System V
ABI, IA32 Supplement.

8.4.2 Process Stack (on entry)
The process stack shall be initialized as described in Chapter 3 of the System V ABI,
IA32 Supplement.

8.4.3 Auxilliary Vector
The auxilliary vector shall be initialized as described in Chapter 3 of the System V
ABI, IA32 Supplement.

8.4.4 Environment
There are no architecture specific requirements for environment initialization.
8.5 Coding Examples

8.5.1 Introduction
LSB-conforming applications may follow the coding examples provided in chapter 3 of the System V ABI, IA32 Supplement in order to implement certain fundamental operations.

8.5.2 Code Model Overview/Architecture Constraints
Chapter 3 of the System V ABI, IA32 Supplement provides an overview of the code model.

8.5.3 Position-Independent Function Prologue
LSB-conforming applications using position independent functions may use the techniques described in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.4 Data Objects
LSB-conforming applications accessing non-stack resident data objects may do so as described in Chapter 3 of the System V ABI, IA32 Supplement, including both absolute and position independent data access techniques.

8.5.5 Function Calls
8.5.5.1 Absolute Direct Function Call
LSB-conforming applications using direct function calls with absolute addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.5.2 Absolute Indirect Function Call
LSB-conforming applications using indirect function calls with absolute addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.5.3 Position-Independent Direct Function Call
LSB-conforming applications using direct function calls with position independent addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.5.4 Position-Independent Indirect Function Call
LSB-conforming applications using indirect function calls with position independent addressing may follow the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.5.6 Branching
LSB-conforming applications may follow the branching examples given in Chapter 3 of the System V ABI, IA32 Supplement.
8.6 C Stack Frame

8.6.1 Variable Argument List

As described in Chapter 3 of the System V ABI, IA32 Supplement, LSB-conforming applications using variable argument lists shall use the facilities defined in the header file `<stdarg.h>` to deal with variable argument lists.

**Note:** This is a requirement of ISO C (1999) and ISO POSIX (2003) as well as System V ABI, IA32 Supplement.

8.6.2 Dynamic Allocation of Stack Space

LSB-conforming applications may allocate space using the stack following the examples given in Chapter 3 of the System V ABI, IA32 Supplement.

8.7 Debug Information

There are no architecture specific requirements for debugging information for this architecture. LSB-conforming applications may utilize DWARF sections as described in the generic specification.
9 Object Format

9.1 Introduction

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

9.2 ELF Header

9.2.1 Machine Information

LSB-conforming applications shall use the Machine Information as defined in Chapter 4 of the System V ABI, IA32 Supplement, including the e_ident array members for EI_CLASS and EI_DATA, the processor identification in e_machine and flags in e_flags. The operating system identification field, in e_ident[EI_OSABI] shall be ELFOSABI_NONE (0).

9.3 Special Sections

9.3.1 Special Sections

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

9.3.1.1 ELF Special Sections

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.got</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_WRITE</td>
</tr>
<tr>
<td>.plt</td>
<td>SHT_PROGBITS</td>
<td>SHF_ALLOC+SHF_EXE CINSTR</td>
</tr>
</tbody>
</table>

.got

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

.plt

This section holds the procedure linkage table.

9.3.1.2 Addition Special Sections

The following additional sections are defined here.
Table 9-2 Additional Special Sections

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.rel.dyn</td>
<td>SHT_REL</td>
<td>SHF_ALLOC</td>
</tr>
</tbody>
</table>

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

9.4 Symbol Table

LSB-conforming applications shall use the Symbol Table section as defined in Chapter 4 of the System V ABI, IA32 Supplement.

9.5 Relocation

9.5.1 Introduction

LSB-conforming implementations shall support Relocation as defined in Chapter 4 of the System V ABI, IA32 Supplement and as described below.

9.5.2 Relocation Types

The relocation types described in Chapter 4 of the System V ABI, IA32 Supplement shall be supported.
10 Program Loading and Dynamic Linking

10.1 Introduction

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

10.2 Program Header

10.2.1 Introduction

As described in System V ABI Update, the program header is an array of structures, each describing a segment or other information the system needs to prepare the program for execution.

10.2.2 Types

The IA32 architecture does not define any additional program header types beyond those required in the generic LSB Core specification.

10.2.3 Flags

The IA32 architecture does not define any additional program header flags beyond those required in the generic LSB Core specification.

10.3 Program Loading

LSB-conforming systems shall support program loading as defined in Chapter 5 of the System V ABI, IA32 Supplement.

10.4 Dynamic Linking

LSB-conforming systems shall support dynamic linking as defined in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.1 Dynamic Section

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

DT_PLTGOT

On the Intel386 architecture, this entrys d_ptr member gives the address of the first entry in the global offset table.

10.4.2 Global Offset Table

LSB-conforming implementations shall support use of the global offset table as described in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.3 Shared Object Dependencies

There are no architecture specific requirements for shared object dependencies; see the generic LSB-Core specification.
10.4.4 Function Addresses
Function addresses shall behave as specified in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.5 Procedure Linkage Table
LSB-conforming implementations shall support a Procedure Linkage Table as described in Chapter 5 of the System V ABI, IA32 Supplement.

10.4.6 Initialization and Termination Functions
There are no architecture specific requirements for initialization and termination functions; see the generic LSB-Core specification.
III Base Libraries
11 Libraries

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

Interfaces that are unique to the IA32 platform are defined here. This section should be used in conjunction with the corresponding section in the Linux Standard Base Specification.

11.1 Program Interpreter/Dynamic Linker

The Program Interpreter shall be /lib/ld-1sb.so.3.

11.2 Interfaces for libc

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

<table>
<thead>
<tr>
<th>Library:</th>
<th>libc</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libc.so.6</td>
</tr>
</tbody>
</table>

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

[LFS] Large File Support
[LSB] This Specification
[SUSv2] SUSv2
[SVID.3] SVID Issue 3
[SVID.4] SVID Issue 4

11.2.1 RPC

11.2.1.1 Interfaces for RPC

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

<table>
<thead>
<tr>
<th>authnone_create(GLIBC_2.0) [SVID.4]</th>
<th>clnt_create(GLIBC_2.0) [SVID.4]</th>
<th>clnt_pcreateerror(GLIBC_2.0) [SVID.4]</th>
<th>clnt_errno(GLIBC_2.0) [SVID.4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>clnt_perror(GLIBC_2.0) [SVID.4]</td>
<td>clnt_spcreateerror(GLIBC_2.0) [SVID.4]</td>
<td>clnt_sperror(GLIBC_2.0) [SVID.4]</td>
<td>clnt_sperror(GLIBC_2.0) [SVID.4]</td>
</tr>
<tr>
<td>key_decryptsession(GLIBC_2.1) [SVID.3]</td>
<td>pmap_getport(GLIBC_2.0) [LSB]</td>
<td>pmap_set(GLIBC_2.0) [LSB]</td>
<td>pmap_unset(GLIBC_2.0) [LSB]</td>
</tr>
<tr>
<td>svc_getreqset(GLIB)</td>
<td>svc_register(GLIB)</td>
<td>svc_run(GLIBC_2.0)</td>
<td>svc_sendreply(GLIB)</td>
</tr>
</tbody>
</table>
11 Libraries

36

BC_2.0) [SVID.3]
C_2.0) [LSB]
0) [LSB]
IBC_2.0) [LSB]
svcerr_auth(GLIBC_2.0) [SVID.3]
svcerr_decode(GLIBC_2.0) [SVID.3]
svcerr_noproc(GLIBC_2.0) [SVID.3]
svcerr_noprog(GLIBC_2.0) [SVID.3]
svcerr_progvers(GLIBC_2.0) [SVID.3]
svcerr_systemerr(GLIBC_2.0) [SVID.3]
svcerr_weakauth(GLIBC_2.0) [SVID.3]
svctcp_create(GLIBC_2.0) [SVID.3]
svcudp_create(GLIBC_2.0) [SVID.3]
xdr_accepted_repl(y(GLIBC_2.0) [SVID.3]
xdr_array(GLIBC_2.0) [SVID.3]
xdr Bool(GLIBC_2.0) [SVID.3]
xdr_bytes(GLIBC_2.0) [SVID.3]
xdr_callhdr(GLIBC_2.0) [SVID.3]
xdr_callmsg(GLIBC_2.0) [SVID.3]
xdr_char(GLIBC_2.0) [SVID.3]
xdr_double(GLIBC_2.0) [SVID.3]
xdr_enum(GLIBC_2.0) [SVID.3]
xdr_float(GLIBC_2.0) [SVID.3]
xdr_free(GLIBC_2.0) [SVID.3]
xdr_int(GLIBC_2.0) [SVID.3]
xdr_long(GLIBC_2.0) [SVID.3]
xdr_opaque(GLIBC_2.0) [SVID.3]
xdr_opaque_auth(GLIBC_2.0) [SVID.3]
xdr_pointer(GLIBC_2.0) [SVID.3]
xdr_reference(GLIBC_2.0) [SVID.3]
xdr_rejected_repl(y(GLIBC_2.0) [SVID.3]
xdr_replymsg(GLIBC_2.0) [SVID.3]
xdr_short(GLIBC_2.0) [SVID.3]
xdr_string(GLIBC_2.0) [SVID.3]
xdr_u_char(GLIBC_2.0) [SVID.3]
xdr_u_int(GLIBC_2.0) [LSB]
xdr_u_long(GLIBC_2.0) [SVID.3]
xdr_u_short(GLIBC_2.0) [SVID.3]
xdr_union(GLIBC_2.0) [SVID.3]
xdr_vector(GLIBC_2.0) [SVID.3]
xdr_void(GLIBC_2.0) [SVID.3]
xdr_wrapstring(GLIBC_2.0) [SVID.3]
xdrmem_create(GLIBC_2.0) [SVID.3]
xdrrec_create(GLIBC_2.0) [SVID.3]
xdrrec_eof(GLIBC_2.0) [SVID.3]

11.2.2 System Calls

11.2.2.1 Interfaces for System Calls

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.

Table 11-3 libc - System Calls Function Interfaces

<table>
<thead>
<tr>
<th>__fxstat(GLIBC_2.0) [LSB]</th>
<th>__getpgid(GLIBC_2.0) [LSB]</th>
<th>__lxstat(GLIBC_2.0) [LSB]</th>
<th>__xmknod(GLIBC_2.0) [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>__xstat(GLIBC_2.0) [LSB]</td>
<td>access(GLIBC_2.0) [SUSv3]</td>
<td>acct(GLIBC_2.0) [LSB]</td>
<td>alarm(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>brk(GLIBC_2.0) [SUSv2]</td>
<td>chdir(GLIBC_2.0) [SUSv3]</td>
<td>chmod(GLIBC_2.0) [SUSv3]</td>
<td>chown(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>chroot(GLIBC_2.0)</td>
<td>clock(GLIBC_2.0)</td>
<td>close(GLIBC_2.0)</td>
<td>closedir(GLIBC_2.0)</td>
</tr>
<tr>
<td>Function</td>
<td>SUSv2</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>creat(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dup(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>dup2(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>execl(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>execp(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>execv(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>execvp(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>exit(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fchdir(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fchown(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fork(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fstatvfs(GLIBC_2.1)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fcntl(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>fdatasync(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>flock(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ftruncate(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getcontext(GLIBC_2.1)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getgroupid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getgroups(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getitimer(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getpagesize(GLIBC_2.0)</td>
<td>SUSv2</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getpgid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getppid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getpriority(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getrlimit(GLIBC_2.2)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getrusage(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getsid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getuid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>getwd(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>initgroups(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ioctl(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>kill(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>killpg(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>lchown(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>link(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>lockf(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>lseek(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mkdir(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mkfifo(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mlock(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>mlockall(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>munlock(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>munlockall(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>nanosleep(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>nice(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>open(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pipe(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pathconf(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>pause(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>poll(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>read(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>readdir(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>readdir_r(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_get_priority_max</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_get_priority_min</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_getparam</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_setscheduler</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_getidot</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_rr_get_interval</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_setparam</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sched_setscheduler</td>
<td>SUSv3</td>
<td>SUSv3</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>
11 Libraries

<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>Function</th>
<th>SUSv3</th>
<th>Function</th>
<th>SUSv3</th>
<th>Function</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>er(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>rval(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>r(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>sched_yield(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>select(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>setcontext(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>setegid(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>setgid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>setitimer(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>setpriority(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>setpgid(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>setrlimit(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>setreuid(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>setrlimit64(GLIBC_2.1)</td>
<td>SUSv3</td>
<td>setregid(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>setsid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>setuid(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>statvfs(GLIBC_2.1)</td>
<td>SUSv3</td>
<td>symlink(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>time(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>times(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ulimit(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>utime(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>uname(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>utimes(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>vfork(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>write(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>writev(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>wait(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>wait4(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>waitpid(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>write(GLIBC_2.0)</td>
<td>SUSv3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.2.3 Standard I/O

11.2.3.1 Interfaces for Standard I/O

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.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>Function</th>
<th>SUSv3</th>
<th>Function</th>
<th>SUSv3</th>
<th>Function</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IO_feof(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>_IO_getc(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>_IO_putc(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>_IO_puts(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_IO_geteof(GLIBC_2.0)</td>
<td>SUSv3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asprintf(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>clearerr(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>ctermid(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fclose(GLIBC_2.1)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdopen(GLIBC_2.1)</td>
<td>SUSv3</td>
<td>feof(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>ferror(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fflush(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fflush_unlocked(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fgetc(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fgetpos(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fgets(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetwc_unlocked(GLIBC_2.2)</td>
<td>SUSv3</td>
<td>fileno(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>flockfile(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fopen(GLIBC_2.1)</td>
<td>SUSv3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fprintf(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fputc(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fputs(GLIBC_2.0)</td>
<td>SUSv3</td>
<td>fwrite(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>
### 11 Libraries

<table>
<thead>
<tr>
<th>Function</th>
<th>Symbol</th>
<th>Standard</th>
<th>Function</th>
<th>Symbol</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>freopen(GLIBC_2.0)</td>
<td>fopen(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>fscanf(GLIBC_2.0)</td>
<td>fscanf(GLIBC_2.0)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>fsetpos(GLIBC_2.2)</td>
<td>fseek(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>ftell(GLIBC_2.0)</td>
<td>ftell(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getc(GLIBC_2.0)</td>
<td>getc_unlocked(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>getchar(GLIBC_2.0)</td>
<td>getchar_unlocked(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>getw(GLIBC_2.0)</td>
<td>pclose(GLIBC_2.1)</td>
<td>[SUSv2]</td>
<td>popen(GLIBC_2.1)</td>
<td>popen(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>putc(GLIBC_2.0)</td>
<td>putc_unlocked(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>putchar(GLIBC_2.0)</td>
<td>putchar_unlocked(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>puts(GLIBC_2.0)</td>
<td>putw(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>remove(GLIBC_2.0)</td>
<td>remove(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>rewinddir(GLIBC _2.0)</td>
<td>scanf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>seekdir(GLIBC_2.0)</td>
<td>seekdir(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setbuffer(GLIBC_2.0)</td>
<td>setvbuf(GLIBC_2.0)</td>
<td>[LSB]</td>
<td>snprintf(GLIBC_2.0)</td>
<td>snprintf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>sscanf(GLIBC_2.0)</td>
<td>telldir(GLIBC_2.0)</td>
<td>[LSB]</td>
<td>tempnam(GLIBC_2.0)</td>
<td>tempnam(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>vsprintf(GLIBC_2.0)</td>
<td>vsprintf(GLIBC_2.0)</td>
<td>[LSB]</td>
<td>vfprintf(GLIBC_2.0)</td>
<td>vfprintf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>vsprintf(GLIBC_2.0)</td>
<td>vfprintf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>fprintf(GLIBC_2.0)</td>
<td>fprintf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

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.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Symbol</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>stderr(GLIBC_2.0)</td>
<td>stderr(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>stdin(GLIBC_2.0)</td>
<td>stdin(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>stdout(GLIBC_2.0)</td>
<td>stdout(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

### 11.2.4 Signal Handling

#### 11.2.4.1 Interfaces for Signal Handling

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.

#### Table 11-6 libc - Signal Handling Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Symbol</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>_<em>libc_current</em> sig rtmax(GLIBC_2.1)</td>
<td>__libc_current_sigtmax(GLIBC_2.1)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>_<em>libc_current</em> sig rtmin(GLIBC_2.1)</td>
<td>__sigsetjmp(GLIBC_2.0)</td>
<td>[LSB]</td>
</tr>
<tr>
<td>__sysv_signal(GLIBC_2.0)</td>
<td>__sysv_signal(GLIBC_2.0)</td>
<td>[LSB]</td>
</tr>
</tbody>
</table>
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.

Table 11-7 libc - Signal Handling Data Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>_sys_siglist(GLIBC_2.3.3)</td>
<td></td>
<td>[LSB]</td>
</tr>
</tbody>
</table>

11.2.5 Localization Functions

11.2.5.1 Interfaces for Localization Functions

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.

Table 11-8 libc - Localization Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>SUSv3</th>
<th>LSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind_textdomain_codeset(GLIBC_2.2)</td>
<td></td>
<td>[LSB]</td>
</tr>
<tr>
<td>bindtextdomain(GLIBC_2.0)</td>
<td></td>
<td>[LSB]</td>
</tr>
<tr>
<td>catclose(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>catgets(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>catopen(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>dcgettext(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>dcngettext(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>dgettext(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>dngettext(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>iconv_open(GLIBC_2.1)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>localeconv(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ngettext(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>nl_langinfo(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>setlocale(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>textdomain(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>
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.

Table 11-9 libc - Localization Functions Data Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>[SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>_nl_msg_cat_cntr(</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glibc_2.0) [LSB]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>[SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept(GLIBC_2.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gethostid(GLIBC_2.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gethostname(GLIBC_2.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Getpeername(GLIBC_2.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connect(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Gethostname(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>If_freenameindex(GLIBC_2.1)</td>
<td>[LSB]</td>
<td></td>
</tr>
<tr>
<td>If_inxtoname(GLIBC_2.1)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>If_nameindex(GLIBC_2.1)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>If_nameindex(GLIBC_2.1)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>If_nameindex(GLIBC_2.1)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>If_nameindex(GLIBC_2.1)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Fgetwc(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Fgetws(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>[SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Btowc(GLIBC_2.0)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Fgetwc(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Fgetws(GLIBC_2.2)</td>
<td></td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Library function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>fputwc(GLIBC_2.2)</code></td>
<td><code>fputwc</code> puts wide-character data into a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>fputws(GLIBC_2.2)</code></td>
<td><code>fputws</code> puts wide-character data into a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>fwide(GLIBC_2.2)</code></td>
<td><code>fwide</code> returns the filewide data type.</td>
<td></td>
</tr>
<tr>
<td><code>fwprintf(GLIBC_2.2)</code></td>
<td><code>fwprintf</code> writes formatted wide-character data to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>fwscanf(GLIBC_2.2)</code></td>
<td><code>fwscanf</code> reads formatted wide-character data from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>getwc(GLIBC_2.2)</code></td>
<td><code>getwc</code> reads a single wide-character from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>getwchar(GLIBC_2.2)</code></td>
<td><code>getwchar</code> reads a single wide-character from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>mblen(GLIBC_2.0)</code></td>
<td><code>mblen</code> returns the length in bytes of a single wide-character.</td>
<td></td>
</tr>
<tr>
<td><code>mbrlen(GLIBC_2.0)</code></td>
<td><code>mbrlen</code> returns the length in bytes of a single wide-character.</td>
<td></td>
</tr>
<tr>
<td><code>mbrtowc(GLIBC_2.0)</code></td>
<td><code>mbrtowc</code> converts a single wide-character to a wide-character.</td>
<td></td>
</tr>
<tr>
<td><code>mbsinit(GLIBC_2.0)</code></td>
<td><code>mbsinit</code> initializes the wide-character locale.</td>
<td></td>
</tr>
<tr>
<td><code>mbsntowcs(GLIBC_2.0)</code></td>
<td><code>mbsntowcs</code> converts a null-terminated wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>mbsrctowcs(GLIBC_2.0)</code></td>
<td><code>mbsrctowcs</code> converts a null-terminated wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>mbsrtowcs(GLIBC_2.0)</code></td>
<td><code>mbsrtowcs</code> converts a null-terminated wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>mbstowcs(GLIBC_2.0)</code></td>
<td><code>mbstowcs</code> converts a wide-character string to a null-terminated wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>mbtowc(GLIBC_2.0)</code></td>
<td><code>mbtowc</code> converts a wide-character to a single wide-character.</td>
<td></td>
</tr>
<tr>
<td><code>putwc(GLIBC_2.2)</code></td>
<td><code>putwc</code> writes a single wide-character to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>putwchar(GLIBC_2.2)</code></td>
<td><code>putwchar</code> writes a single wide-character to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>swprintf(GLIBC_2.2)</code></td>
<td><code>swprintf</code> writes formatted wide-character data to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>swscanf(GLIBC_2.2)</code></td>
<td><code>swscanf</code> reads formatted wide-character data from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>towctrans(GLIBC_2.0)</code></td>
<td><code>towctrans</code> performs wide-character to wide-character transliteration.</td>
<td></td>
</tr>
<tr>
<td><code>towlower(GLIBC_2.0)</code></td>
<td><code>towlower</code> converts a wide-character to lowercase.</td>
<td></td>
</tr>
<tr>
<td><code>towupper(GLIBC_2.0)</code></td>
<td><code>towupper</code> converts a wide-character to uppercase.</td>
<td></td>
</tr>
<tr>
<td><code>ungetwc(GLIBC_2.2)</code></td>
<td><code>ungetwc</code> ungets a wide-character from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>vfwprintf(GLIBC_2.2)</code></td>
<td><code>vfwprintf</code> writes formatted wide-character data to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>vfwscanf(GLIBC_2.2)</code></td>
<td><code>vfwscanf</code> reads formatted wide-character data from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>vswprintf(GLIBC_2.2)</code></td>
<td><code>vswprintf</code> writes formatted wide-character data to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>vswscanf(GLIBC_2.2)</code></td>
<td><code>vswscanf</code> reads formatted wide-character data from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>vwprintf(GLIBC_2.2)</code></td>
<td><code>vwprintf</code> writes formatted wide-character data to a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>vwscanf(GLIBC_2.2)</code></td>
<td><code>vwscanf</code> reads formatted wide-character data from a wide-character stream.</td>
<td></td>
</tr>
<tr>
<td><code>wcpcpy(GLIBC_2.0)</code></td>
<td><code>wcpcpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcpncpy(GLIBC_2.0)</code></td>
<td><code>wcpncpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcrtomb(GLIBC_2.0)</code></td>
<td><code>wcrtomb</code> converts a wide-character to a multi-byte character.</td>
<td></td>
</tr>
<tr>
<td><code>wcscasecmp(GLIBC_2.1)</code></td>
<td><code>wcscasecmp</code> compares two null-terminated wide-character strings case-insensitively.</td>
<td></td>
</tr>
<tr>
<td><code>wcscat(GLIBC_2.0)</code></td>
<td><code>wcscat</code> concatenates two null-terminated wide-character strings.</td>
<td></td>
</tr>
<tr>
<td><code>wcschr(GLIBC_2.0)</code></td>
<td><code>wcschr</code> searches for the first occurrence of a wide-character in a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcscmp(GLIBC_2.0)</code></td>
<td><code>wcscmp</code> compares two null-terminated wide-character strings.</td>
<td></td>
</tr>
<tr>
<td><code>wcscoll(GLIBC_2.0)</code></td>
<td><code>wcscoll</code> compares two null-terminated wide-character strings in the collation order.</td>
<td></td>
</tr>
<tr>
<td><code>wcscpy(GLIBC_2.0)</code></td>
<td><code>wcscpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsnlen(GLIBC_2.1)</code></td>
<td><code>wcsnlen</code> returns the length of a null-terminated wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsncmpeq(GLIBC_2.0)</code></td>
<td><code>wcsncmp</code> compares two null-terminated wide-character strings.</td>
<td></td>
</tr>
<tr>
<td><code>wcsncpy(GLIBC_2.0)</code></td>
<td><code>wcsncpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsnlen(GLIBC_2.1)</code></td>
<td><code>wcsnlen</code> returns the length of a null-terminated wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsnrtombs(GLIBC_2.0)</code></td>
<td><code>wcsnrtombs</code> converts a null-terminated wide-character string to a multi-byte character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcspbrk(GLIBC_2.0)</code></td>
<td><code>wcspbrk</code> searches for the first occurrence of a wide-character in a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsrchr(GLIBC_2.0)</code></td>
<td><code>wcsrchr</code> searches for the last occurrence of a wide-character in a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsrtombs(GLIBC_2.0)</code></td>
<td><code>wcsrtombs</code> converts a null-terminated wide-character string to a multi-byte character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcscasecmp(GLIBC_2.1)</code></td>
<td><code>wcscasecmp</code> compares two null-terminated wide-character strings case-insensitively.</td>
<td></td>
</tr>
<tr>
<td><code>wcscat(GLIBC_2.0)</code></td>
<td><code>wcscat</code> concatenates two null-terminated wide-character strings.</td>
<td></td>
</tr>
<tr>
<td><code>wcscmp(GLIBC_2.0)</code></td>
<td><code>wcscmp</code> compares two null-terminated wide-character strings.</td>
<td></td>
</tr>
<tr>
<td><code>wcscoll(GLIBC_2.0)</code></td>
<td><code>wcscoll</code> compares two null-terminated wide-character strings in the collation order.</td>
<td></td>
</tr>
<tr>
<td><code>wcscpy(GLIBC_2.0)</code></td>
<td><code>wcscpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcsncpy(GLIBC_2.0)</code></td>
<td><code>wcsncpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcstol(GLIBC_2.0)</code></td>
<td><code>wcstol</code> converts a wide-character string to a long integer.</td>
<td></td>
</tr>
<tr>
<td><code>wcstoul(GLIBC_2.0)</code></td>
<td><code>wcstoul</code> converts a wide-character string to a long unsigned integer.</td>
<td></td>
</tr>
<tr>
<td><code>wcstoull(GLIBC_2.1)</code></td>
<td><code>wcstoull</code> converts a wide-character string to a long unsigned long integer.</td>
<td></td>
</tr>
<tr>
<td><code>wcstombs(GLIBC_2.0)</code></td>
<td><code>wcstombs</code> converts a wide-character string to a multi-byte character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcstoumax(GLIBC_2.1)</code></td>
<td><code>wcstoumax</code> converts a wide-character string to a long unsigned long integer.</td>
<td></td>
</tr>
<tr>
<td><code>wcswcs(GLIBC_2.1)</code></td>
<td><code>wcswcs</code> converts a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wcswidth(GLIBC_2.0)</code></td>
<td><code>wcswidth</code> returns the width of a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wctype(GLIBC_2.0)</code></td>
<td><code>wctype</code> determines the type of a wide-character.</td>
<td></td>
</tr>
<tr>
<td><code>wmemcpy(GLIBC_2.0)</code></td>
<td><code>wmemcpy</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wmemmove(GLIBC_2.0)</code></td>
<td><code>wmemmove</code> copies a wide-character string to a wide-character string.</td>
<td></td>
</tr>
<tr>
<td><code>wmemset(GLIBC_2.0)</code></td>
<td><code>wmemset</code> sets a wide-character string to a specified character.</td>
<td></td>
</tr>
<tr>
<td><code>wprintf(GLIBC_2.2)</code></td>
<td><code>wprintf</code> writes formatted wide-character data to a wide-character stream.</td>
<td></td>
</tr>
</tbody>
</table>
11.2.8 String Functions

11.2.8.1 Interfaces for String Functions

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.

Table 11-12 libc - String Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>wscanf(GLIBC_2.0)</td>
<td>[LSB]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

43
11 Libraries

11.2.9 IPC Functions

11.2.9.1 Interfaces for IPC Functions

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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftok(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>msgctl(GLIBC_2.2)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>msgget(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>msgsnd(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>semctl(GLIBC_2.2)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>semget(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>shmat(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>shmdt(GLIBC_2.2)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>shmat(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>shmdt(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>

Table 11-13 libc - IPC Functions Function Interfaces

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>
<thead>
<tr>
<th>Function</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>regcomp(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>regerror(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>regexec(GLIBC_2.4)</td>
<td>LSB</td>
</tr>
<tr>
<td>regfree(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>

Table 11-14 libc - Regular Expressions Function Interfaces

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>
<thead>
<tr>
<th>Function</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>__ctype_get_mb_cur_max(GLIBC_2.0)</td>
<td>LSB</td>
</tr>
<tr>
<td>_tolower(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>_toupper(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isalnum(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isalpha(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isascii(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>iscntrl(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isdigit(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isgraph(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>islower(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isprint(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ispunct(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isspace(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>isupper(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>iswalnum(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
<tr>
<td>iswalpha(GLIBC_2.0)</td>
<td>SUSv3</td>
</tr>
</tbody>
</table>
11 Libraries

iswblank(GLIBC_2.1) [SUSv3]
iswcntrl(GLIBC_2.0) [SUSv3]
iswctype(GLIBC_2.0) [SUSv3]
iswdigit(GLIBC_2.0) [SUSv3]
iswgraph(GLIBC_2.0) [SUSv3]
iswlower(GLIBC_2.0) [SUSv3]
iswprint(GLIBC_2.0) [SUSv3]
iswpunct(GLIBC_2.0) [SUSv3]
iswspace(GLIBC_2.0) [SUSv3]
iswupper(GLIBC_2.0) [SUSv3]
iswxdigit(GLIBC_2.0) [SUSv3]
isxdigit(GLIBC_2.0) [SUSv3]
toascii(GLIBC_2.0) [SUSv3]
tolower(GLIBC_2.0) [SUSv3]
toupper(GLIBC_2.0) [SUSv3]

toascii(GLIBC_2.0) [SUSv3]
tolower(GLIBC_2.0) [SUSv3]
toupper(GLIBC_2.0) [SUSv3]

toascii(GLIBC_2.0) [SUSv3]
tolower(GLIBC_2.0) [SUSv3]
toupper(GLIBC_2.0) [SUSv3]

11.2.12 Time Manipulation

11.2.12.1 Interfaces for Time Manipulation

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.

Table 11-16 libc - Time Manipulation Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjtime(GLIBC_2.0) [LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asctime(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asctime_r(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctime_r(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>difftime(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gmtime(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gmtime_r(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>localtime(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>localtime_r(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mktime(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tzset(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ualarm(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Table 11-17 libc - Time Manipulation Data Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>__daylight(GLIBC_2.0) [LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__timezone(GLIBC_2.0) [LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>__tzname(GLIBC_2.0) [LSB]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>daylight(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>timezone(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tzname(GLIBC_2.0) [SUSv3]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.2.13 Terminal Interface Functions

11.2.13.1 Interfaces for Terminal Interface Functions

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

Table 11-18 libc - Terminal Interface Functions Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfgetispeed(GLIBC)</td>
<td></td>
</tr>
<tr>
<td>cfgetospeed(GLIBC)</td>
<td></td>
</tr>
<tr>
<td>cfmakeraw(GLIBC)</td>
<td></td>
</tr>
<tr>
<td>cfsetispeed(GLIBC)</td>
<td></td>
</tr>
</tbody>
</table>
11 Libraries

<table>
<thead>
<tr>
<th>C.2.0) [SUSv3]</th>
<th>C.2.0) [SUSv3]</th>
<th>C.2.0) [LSB]</th>
<th>C.2.0) [SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfsetspeed(GLIBC_2.0) [SUSv3]</td>
<td>cfsetspeed(GLIBC_2.0) [SUSv3]</td>
<td>tcdrain(GLIBC_2.0) [LSB]</td>
<td>tcdrain(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>tcflush(GLIBC_2.0) [SUSv3]</td>
<td>tcgetattr(GLIBC_2.0) [SUSv3]</td>
<td>tcgetpgrp(GLIBC_2.0) [SUSv3]</td>
<td>tcgetpgrp(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>tcflush(GLIBC_2.0) [SUSv3]</td>
<td>tcsendbreak(GLIBC_2.0) [SUSv3]</td>
<td>tcsetattr(GLIBC_2.0) [SUSv3]</td>
<td>tcsetattr(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>tcflush(GLIBC_2.0) [SUSv3]</td>
<td>tcsetattr(GLIBC_2.0) [SUSv3]</td>
<td>tcsetpgrp(GLIBC_2.0) [SUSv3]</td>
<td>tcsetpgrp(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>tcflush(GLIBC_2.0) [SUSv3]</td>
<td>tcsetattr(GLIBC_2.0) [SUSv3]</td>
<td>tcsetpgrp(GLIBC_2.0) [SUSv3]</td>
<td>tcsetpgrp(GLIBC_2.0) [SUSv3]</td>
</tr>
</tbody>
</table>

11.2.14 System Database Interface

11.2.14.1 Interfaces for System Database Interface

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.

Table 11-19 libc - System Database Interface Function Interfaces

<table>
<thead>
<tr>
<th>endgrent(GLIBC_2.0) [SUSv3]</th>
<th>endprotoent(GLIBC_2.0) [SUSv3]</th>
<th>endpwent(GLIBC_2.0) [SUSv3]</th>
<th>endservent(GLIBC_2.0) [SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>endutent(GLIBC_2.0) [SUSv2]</td>
<td>enduxtrent(GLIBC_2.1) [SUSv3]</td>
<td>getgrent(GLIBC_2.0) [SUSv3]</td>
<td>getgrgid(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>getgrgid_r(GLIBC_2.1.2) [SUSv3]</td>
<td>getgrnam(GLIBC_2.0) [SUSv3]</td>
<td>getgrnam_r(GLIBC_2.1.2) [SUSv3]</td>
<td>getgrouplist(GLIBC_2.2.4) [LSB]</td>
</tr>
<tr>
<td>gethostbyaddr(GLIBC_2.0) [SUSv3]</td>
<td>gethostbyname(GLIBC_2.0) [SUSv3]</td>
<td>getprotobyname(GLIBC_2.0) [SUSv3]</td>
<td>getprotobynumber(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>getprotoent(GLIBC_2.0) [SUSv3]</td>
<td>getpwent(GLIBC_2.0) [SUSv3]</td>
<td>getpwnam(GLIBC_2.0) [SUSv3]</td>
<td>getpwnam_r(GLIBC_2.1.2) [SUSv3]</td>
</tr>
<tr>
<td>getpwuid(GLIBC_2.0) [SUSv3]</td>
<td>getpwuid_r(GLIBC_2.1.2) [SUSv3]</td>
<td>getservbyname(GLIBC_2.0) [SUSv3]</td>
<td>getservbyport(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>getservent(GLIBC_2.0) [SUSv3]</td>
<td>getutent(GLIBC_2.0) [LSB]</td>
<td>getutent_r(GLIBC_2.0) [LSB]</td>
<td>getutxent(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>getutxid(GLIBC_2.1) [SUSv3]</td>
<td>getutxline(GLIBC_2.1) [SUSv3]</td>
<td>pututxline(GLIBC_2.1) [SUSv3]</td>
<td>setgrent(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>setgroups(GLIBC_2.0) [LSB]</td>
<td>setprotoent(GLIBC_2.0) [SUSv3]</td>
<td>setpwent(GLIBC_2.0) [SUSv3]</td>
<td>setservent(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>setutent(GLIBC_2.0) [LSB]</td>
<td>setutxtent(GLIBC_2.1) [SUSv3]</td>
<td>utmpname(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
</tbody>
</table>

11.2.15 Language Support

11.2.15.1 Interfaces for Language Support

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.
11 Libraries

11.2.16 Large File Support

11.2.16.1 Interfaces for Large File Support

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.

Table 11-21 libc - Large File Support Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>LFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>__fxstat64(GLIBC_2.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__lstat64(GLIBC_2.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>__xfstat64(GLIBC_2.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>creat64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fgetpos64(GLIBC_2.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fopen64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fstatvfs64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ftell64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ftw64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getrlimit64(GLIBC_2.2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mmap64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>memset64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmpfile64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>truncate64(GLIBC_2.1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.2.17 Standard Library

11.2.17.1 Interfaces for Standard Library

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.

Table 11-22 libc - Standard Library Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>LSB</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Exit(GLIBC_2.1.1)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>_assert_fail(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>_cxa_atexit(GLIBC_2.1.3)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>_errno_location(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>__fpending(GLIBC_2.2)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>__getpagesize(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>__isinf(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>__isinf(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>__isnfl(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>__sysconf(GLIBC_2.2)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>abort(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>abs(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>atof(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>atoi(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>atol(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>atoll(GLIBC_2.0)</td>
<td></td>
<td>SUSv3</td>
</tr>
<tr>
<td>basename(GLIBC)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Function</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>bsearch(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>calloc(GLIBC_2.0)</td>
</tr>
<tr>
<td>cuserid(GLIBC_2.0)</td>
<td>[SUSv2]</td>
<td>daemonic(GLIBC_2.0)</td>
</tr>
<tr>
<td>drand48(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>ecvt(GLIBC_2.0)</td>
</tr>
<tr>
<td>error(GLIBC_2.0)</td>
<td>[LSB]</td>
<td>errorand48(GLIBC_2.0)</td>
</tr>
<tr>
<td>fnmatch(GLIBC_2.2.3)</td>
<td>[SUSv3]</td>
<td>fpathconf(GLIBC_2.0)</td>
</tr>
<tr>
<td>gctv(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>getaddrinfo(GLIBC_2.0)</td>
</tr>
<tr>
<td>getenv(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>getlogin(GLIBC_2.0)</td>
</tr>
<tr>
<td>getopt(GLIBC_2.0)</td>
<td>[LSB]</td>
<td>getopt_long(GLIBC_2.0)</td>
</tr>
<tr>
<td>gettimeofday(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>glob(GLIBC_2.0)</td>
</tr>
<tr>
<td>globfree64(GLIBC_2.1)</td>
<td>[LSB]</td>
<td>grantpt(GLIBC_2.0)</td>
</tr>
<tr>
<td>hsearch(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>htonl(GLIBC_2.0)</td>
</tr>
<tr>
<td>imaxdiv(GLIBC_2.1.1)</td>
<td>[SUSv3]</td>
<td>inet_addr(GLIBC_2.0)</td>
</tr>
<tr>
<td>inet_pton(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>initstate(GLIBC_2.0)</td>
</tr>
<tr>
<td>isblank(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>insque(GLIBC_2.0)</td>
</tr>
<tr>
<td>longjmp(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>ldiv(GLIBC_2.0)</td>
</tr>
<tr>
<td>lldiv(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>longjmp(GLIBC_2.0)</td>
</tr>
<tr>
<td>malloc(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>memmem(GLIBC_2.0)</td>
</tr>
<tr>
<td>mktemp(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>mkstemp(GLIBC_2.0)</td>
</tr>
<tr>
<td>ntohl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>openlog(GLIBC_2.0)</td>
</tr>
<tr>
<td>perror(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>seterr(GLIBC_2.0)</td>
</tr>
</tbody>
</table>
11 Libraries

<table>
<thead>
<tr>
<th>Function</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>posix_memalign(GLIBC_2.2) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>posix_openpt(GLIBC_2.2.1) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>ptsname(GLIBC_2.1) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>putenv(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>qsort(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>rand(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>rand_r(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>random(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>realloc(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>realpath(GLIBC_2.3) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>remque(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>seed48(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>setenv(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sethostname(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>setlogmask(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>setstate(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>srand(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>srand48(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>srand48(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>srandom(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>strtod(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>strtol(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>strtoul(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>swapcontext(GLIBC_2.1) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>syslog(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>system(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>tdelete(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>tfind(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>tmpfile(GLIBC_2.1) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>tsearch(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>ttyname(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>ttyname_r(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>twalk(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>unlockpt(GLIBC_2.1) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>unsetenv(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>usleep(GLIBC_2.0) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>vscanf(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>vsscanf(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>vsyslog(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>warn(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>warnx(GLIBC_2.0) [LSB]</td>
<td></td>
</tr>
<tr>
<td>wordexp(GLIBC_2.1) [SUSv3]</td>
<td></td>
</tr>
<tr>
<td>wordfree(GLIBC_2.1) [SUSv3]</td>
<td></td>
</tr>
</tbody>
</table>

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.

Table 11-23 libc - Standard Library Data Interfaces

<table>
<thead>
<tr>
<th>__environ(GLIBC_2.0) [LSB]</th>
<th>__environ(GLIBC_2.0) [LSB]</th>
<th>__sys_errlist(GLIBC_2.3) [LSB]</th>
<th>environ(GLIBC_2.0) [SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>getdate_err(GLIBC_2.1) [SUSv3]</td>
<td>optarg(GLIBC_2.0) [SUSv3]</td>
<td>opterr(GLIBC_2.0) [SUSv3]</td>
<td>optarg(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>optarg(GLIBC_2.0) [SUSv3]</td>
<td>optarg(GLIBC_2.0) [SUSv3]</td>
<td>optarg(GLIBC_2.0) [SUSv3]</td>
<td>optarg(GLIBC_2.0) [SUSv3]</td>
</tr>
</tbody>
</table>

11.3 Data Definitions for libc

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

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

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

11.3.1 arpa/inet.h

extern uint32_t htonl(uint32_t);
extern uint16_t htons(uint16_t);
extern in_addr_t inet_addr(const char *);
extern char *inet_ntoa(struct in_addr);
extern const char *inet_ntop(int, const void *, char *, socklen_t);
extern int inet_pton(int, const char *, void *);
extern uint32_t ntohl(uint32_t);
extern uint16_t ntohs(uint16_t);

11.3.2 assert.h

extern void __assert_fail(const char *, const char *, unsigned int,
const char *);

11.3.3 ctype.h

extern int _tolower(int);
extern int _toupper(int);
extern int isalnum(int);
extern int isalpha(int);
extern int isascii(int);
extern int iscntrl(int);
extern int isdigit(int);
extern int isgraph(int);
extern int islower(int);
extern int isprint(int);
extern int ispunct(int);
extern int isspace(int);
extern int isupper(int);
extern int isxdigit(int);
extern int toascii(int);
extern int tolower(int);
extern int toupper(int);
extern int isblank(int);
extern const unsigned short **__ctype_b_loc(void);
extern const int32_t **__ctype_toupper_loc(void);
extern const int32_t **__ctype_tolower_loc(void);

11.3.4 dirent.h

extern void rewinddir(DIR *);
extern void seekdir(DIR *, long int);
extern long int telldir(DIR *);
extern int closedir(DIR *);
extern DIR *opendir(const char *);
extern struct dirent *readdir(DIR *);
extern struct dirent64 *readdir64(DIR *);
extern int readdir_r(DIR *, struct dirent *, struct dirent **);

11.3.5 err.h

extern void err(int, const char *, ...);
extern void errx(int, const char *, ...);
extern void warn(const char *, ...);
extern void warnx(const char *, ...);
extern void error(int, int, const char *, ...);

11.3.6 errno.h

#define EDEADLOCK EDEADLK

extern int *__errno_location(void);

11.3.7 fcntl.h

#define F_GETLK64 12
#define F_SETLK64 13
#define F_SETLKW64 14

extern int lockf64(int, int, off64_t);
extern int fcntl(int, int, ...);

11.3.8 fmtmsg.h

extern int fmtmsg(long int, const char *, int, const char *, const char *, const char *);

11.3.9 fnmatch.h

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

11.3.10 ftw.h

extern int ftw(const char *, __ftw_func_t, int);
extern int ftw64(const char *, __ftw64_func_t, int);
extern int nftw(const char *, __nftw_func_t, int, int);
extern int nftw64(const char *, __nftw64_func_t, int, int);

11.3.11 getopt.h

extern int getopt_long(int, char *const, const char *,
const struct option *, int *);
extern int getopt_long_only(int, char *const, const char *,
const struct option *, int *);
11.3.12 glob.h

extern int glob(const char *, int, int (*__errfunc) (const char *p1, int p2), glob_t *);
extern int glob64(const char *, int, int (*__errfunc) (const char *p1, int p2), glob64_t *);
extern void globfree(glob_t *);
extern void globfree64(glob64_t *);

11.3.13 grp.h

extern void endgrent(void);
extern struct group *getgrent(void);
extern struct group *getgrgid(gid_t);
extern struct group *getgrnam(char *);
extern int initgroups(const char *, gid_t);
extern void setgrent(void);
extern int setgroups(size_t, const gid_t *);
extern int getgrgid_r(gid_t, struct group *, char *, size_t, struct group **);
extern int getgrnam_r(const char *, struct group *, char *, size_t, struct group **);
extern int getgrouplist(const char *, gid_t, gid_t *, int *);

11.3.14 iconv.h

extern size_t iconv(iconv_t, char **, size_t *, char **, size_t *);
extern int iconv_close(iconv_t);
extern iconv_t iconv_open(char *, char *);

11.3.15 inttypes.h

typedef long long int intmax_t;
typedef unsigned int uintptr_t;
typedef unsigned long long int uintmax_t;
typedef unsigned long long int uint64_t;

extern intmax_t strtoimax(const char *, char **, int);
extern uintmax_t strtoumax(const char *, char **, int);
extern intmax_t wcstoimax(const wchar_t *, wchar_t **, int);
extern intmax_t wcstoumax(const wchar_t *, wchar_t **, int);
extern intmax_t imaxabs(intmax_t);
extern imaxdiv_t imaxdiv(intmax_t, intmax_t);

11.3.16 langinfo.h

extern char *nl_langinfo(nl_item);

11.3.17 libgen.h

extern char *basename(const char *);
extern char *dirname(char *);
11.3.18 libintl.h

extern char *bindtextdomain(const char *, const char *);
extern char *dcgettext(const char *, const char *, int);
extern char *dgettext(const char *, const char *);
extern char *gettext(const char *);
extern char *textdomain(const char *);
extern char *bind_textdomain_codeset(const char *, const char *);
extern char *dcngettext(const char *, const char *, const char *,
unsigned long int, int);
extern char *dngettext(const char *, const char *, const char *,
unsigned long int);
extern char *ngettext(const char *, const char *, unsigned long int);

11.3.19 limits.h

#define LONG_MAX        0x7FFFFFFFL
#define ULONG_MAX       0xFFFFFFFFUL
#define CHAR_MAX        SCHAR_MAX
#define CHAR_MIN        SCHAR_MIN
#define PTHREAD_STACK_MIN       16384

11.3.20 locale.h

extern struct lconv *localeconv(void);
extern char *setlocale(int, const char *);
extern locale_t uselocale(locale_t);
extern void freelocale(locale_t);
extern locale_t duplocale(locale_t);
extern locale_t newlocale(int, const char *, locale_t);

11.3.21 monetary.h

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

11.3.22 net/if.h

extern void if_freenameindex(struct if_nameindex *);
extern char *if_indextoname(unsigned int, char *);
extern locale_t uselocale(locale_t);
extern void freelocale(locale_t);
extern locale_t duplocale(locale_t);
extern locale_t newlocale(int, const char *, locale_t);

11.3.23 netdb.h

extern void endprotoent(void);
extern void endservent(void);
extern void freeaddrinfo(struct addrinfo *);
extern const char *gai_strerror(int);
extern int getaddrinfo(const char *, const char *,
const struct addrinfo *,
struct addrinfo **);
extern struct hostent *gethostbyaddr(const void *, socklen_t, int);
extern struct hostent *gethostbyname(const char *);
extern struct protoent *getprotobyname(const char *);
extern struct protoent *getprotobynumber(int);
extern struct protoent *getprotoent(void);
extern struct servent *getservbyname(const char *, const char *);
extern struct servent *getservbyport(int, const char *);
extern struct servent *getservent(void);
extern void setprotoent(int);
extern void setservent(int);
extern int *__h_errno_location(void);

11.3.24 netinet/in.h

extern int bindresvport(int, struct sockaddr_in *);

11.3.25 netinet/ip.h

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

11.3.26 netinet/tcp.h

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

11.3.27 netinet/udp.h

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

11.3.28 nl_types.h

extern int catclose(nl_catd);
extern char *catgets(nl_catd, int, int, const char *);
extern nl_catd catopen(const char *, int);

11.3.29 poll.h

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

11.3.30 pty.h

extern int openpty(int *, int *, char *, struct termios *,
                   struct winsize *);
extern int forkpty(int *, char *, struct termios *, struct winsize *);

11.3.31 pwd.h

extern void endpwent(void);
extern struct passwd *getpwent(void);
extern struct passwd *getpwnam(char *);
extern struct passwd *getpwnam_r(char *, struct passwd *, char *, size_t, struct passwd **);
extern void setpwent(void);
extern int getpwuid(uid_t, struct passwd **);
extern int getpwnam_r(char *, struct passwd *, char *, size_t, struct passwd **);

11.3.32 regex.h

extern int regcomp(regex_t *, const char *, int);
extern size_t regerror(int, const regex_t *, char *, size_t);
extern int regexec(const regex_t *, const char *, size_t, regmatch_t, int);
extern void regfree(regex_t *);

11.3.33 rpc/auth.h

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

11.3.34 rpc/clnt.h

extern struct CLIENT *clnt_create(const char *, const u_long, const u_long,
                                  const char *);
extern void clnt_pcreateerror(const char *);
extern void clnt_perror(enum clnt_stat);
extern void clnt_perror(struct CLIENT *, const char *);
extern char *clnt_sperror(enum clnt_stat);
extern char *clnt_sperror(struct CLIENT *, const char *);

11.3.35 rpc/pmap_clnt.h

extern u_short pmap_getport(struct sockaddr_in *, const u_long, const u_long, u_int);
extern bool_t pmap_set(const u_long, const u_long, int, u_short);
extern bool_t pmap_unset(u_long, u_long);

11.3.36 rpc/rfcmsg.h

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

11.3.37 rpc/svc.h

extern void svc_getreqset(fd_set *);
extern bool_t svc_register(SVCXPRT *, rpcprog_t, rpcvers_t, __dispatch_fn_t, rpcprot_t);
extern void svc_run(void);
extern bool_t svc_sendreply(SVCXPRT *, xdrproc_t, caddr_t);
extern void svcerr_auth(SVCXPRT *, enum auth_stat);
extern void svcerr_decode(SVCXPRT *);
extern void svcerr_noproc(SVCXPRT *);
extern void svcerr_noprog(SVCXPRT *);
11 Libraries

extern void svcerr_progvers(SVCXPRT *, rpcvers_t, rpcvers_t);
extern void svcerr_systemerr(SVCXPRT *);
extern void svcerr_weakauth(SVCXPRT *);
extern SVCXPRT *svctcp_create(int, u_int, u_int);
extern SVCXPRT *svcudp_create(int);

11.3.38 rpc/types.h

/*
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

11.3.39 rpc/xdr.h

extern bool_t xdr_array(XDR *, caddr_t *, u_int *, u_int, u_int,
            xdrproc_t);
extern bool_t xdr_bool(XDR *, bool_t *);
extern bool_t xdr_bytes(XDR *, char **, u_int *, u_int);
extern bool_t xdr_char(XDR *, char *);
extern bool_t xdr_double(XDR *, double *);
extern bool_t xdr_enum(XDR *, enum_t *);
extern bool_t xdr_float(XDR *, float *);
extern void xdr_free(xdrproc_t, char *);
extern bool_t xdr_int(XDR *, int *);
extern bool_t xdr_long(XDR *, long int *);
extern bool_t xdrOpaque(XDR *, caddr_t, u_int);
extern bool_t xdr_pointer(XDR *, char **, u_int, xdrproc_t);
extern bool_t xdr_reference(XDR *, caddr_t *, u_int, xdrproc_t);
extern bool_t xdr_short(XDR *, short *);
extern bool_t xdr_string(XDR *, char **, u_int);
extern bool_t xdr_u_char(XDR *, u_char *);
extern bool_t xdr_u_int(XDR *, u_int *);
extern bool_t xdr_u_long(XDR *, u_long *);
extern bool_t xdr_u_short(XDR *, u_short *);
extern bool_t xdr_union(XDR *, enum_t *, char *,
            const struct xdr_discrim *, xdrproc_t);
extern bool_t xdr_vector(XDR *, char *, u_int, u_int, xdrproc_t);
extern bool_t xdr_void(void);
extern bool_t xdr_wrapstring(XDR *, char *);
extern void xdrmem_create(XDR *, caddr_t, u_int, enum xdr_op);
extern void xdrrec_create(XDR *, u_int, u_int, caddr_t,
        int (*__readit) (char *p1, char *p2, int p3)
        , int (*__writeit) (char *p1, char *p2, int p3) p3);
extern typedef int bool_t xdrrec_eof(XDR *);

11.3.40 sched.h

extern int sched_get_priority_max(int);
extern int sched_get_priority_min(int);
extern int sched_getparam(pid_t, struct sched_param *);
extern int sched_setscheduler(pid_t);
extern int sched_rr_get_interval(pid_t, struct timespec *);
extern int sched_setparam(pid_t, const struct sched_param *);
extern int sched_setscheduler(pid_t, int, const struct sched_param *);
extern int sched_yield(void);
11 Libraries

11.3.41 search.h

extern int hcreate(size_t);
extern ENTRY *hsearch(ENTRY, ACTION);
extern void insque(void *, void *);
extern int hcreate(void *, ENTRY, ACTION);
extern ENTRY *hsearch(ENTRY, ACTION);
extern void *lfind(const void *, const void *, size_t *, size_t, __compar_fn_t);
extern void *lsearch(const void *, void *, size_t *, size_t, __compar_fn_t);
extern void remque(void *);
extern void hdestroy(void);
extern void *tdelete(const void *, void **, __compar_fn_t);
extern void *tfind(const void *, void *const *, __compar_fn_t);
extern void *tsearch(const void *, void **, __compar_fn_t);
extern void twalk(const void *, __action_fn_t);

11.3.42 setjmp.h

typedef int __jmp_buf[6];
extern int __sigsetjmp(jmp_buf, int);
extern void longjmp(jmp_buf, int);
extern void siglongjmp(sigjmp_buf, int);
extern void _longjmp(jmp_buf, int);
extern int _setjmp(jmp_buf);

11.3.43 signal.h

#define SIGEV_PAD_SIZE ((SIGEV_MAX_SIZE/sizeof(int))-3)
#define SI_PAD_SIZE ((SI_MAX_SIZE/sizeof(int))-3)
struct sigaction {
    union {
        sighandler_t _sa_handler;
        void (*_sa_sigaction) (int, siginfo_t *, void *);
    } __sigaction_handler;
    sigset_t sa_mask;
    unsigned long int sa_flags;
    void (*sa_restorer) (void);
};
#define MINSIGSTKSZ 2048
#define SIGSTKSZ 8192
struct _fpreg {
    unsigned short significand[4];
    unsigned short exponent;
};
struct _fpreg {
    unsigned short significand[4];
    unsigned short exponent;
    unsigned short padding[3];
};
struct _xmmreg {
    unsigned long int element[4];
};
struct _fpstate {
    unsigned long int cw;
}
11 Libraries

unsigned long int sw;
unsigned long int tag;
unsigned long int ipoff;
unsigned long int cssel;
unsigned long int dataoff;
unsigned long int dataset;
struct _fpreg _st[8];
unsigned short status;
unsigned short magic;
unsigned long int _fxsr_env[6];
unsigned long int mxcsr;
unsigned long int reserved;
struct _fpxreg _fxsr_st[8];
struct _xmmreg _xmm[8];
unsigned long int padding[56];
}

struct sigcontext {
    unsigned short gs;
    unsigned short __gsh;
    unsigned short fs;
    unsigned short __fsh;
    unsigned short es;
    unsigned short __esh;
    unsigned short ds;
    unsigned short __dsh;
    unsigned long int edi;
    unsigned long int esi;
    unsigned long int ebp;
    unsigned long int esp;
    unsigned long int ebx;
    unsigned long int edx;
    unsigned long int ecx;
    unsigned long int eax;
    unsigned long int trapno;
    unsigned long int err;
    unsigned long int eip;
    unsigned short cs;
    unsigned short __csh;
    unsigned long int eflags;
    unsigned long int esp_at_signal;
    unsigned short ss;
    unsigned short __ssh;
    struct _fpstate *fpstate;
    unsigned long int oldmask;
    unsigned long int cr2;
};

extern int __libc_current_sigrtmax(void);
extern int __libc_current_sigrtmin(void);
extern sighandler_t __sysv_signal(int, sighandler_t);
extern char *const _sys_siglist(void);
extern int killpg(pid_t, int);
extern void psignal(int, const char *);
extern int raise(int);
extern int sigaddset(sigset_t *, int);
extern int sigandset(sigset_t *, const sigset_t *, const sigset_t *);
extern int sigdelset(sigset_t *, int);
extern int sigemptyset(sigset_t *);
extern int sigfillset(sigset_t *);
extern int sighold(int);
extern int sigignore(int);
extern int siginterrupt(int, int);
extern int sigismember(const sigset_t *, int);
extern int sigis救命会员(const sigset_t *, int);
11.3.44 stddef.h

typedef unsigned int size_t;
typedef int ptrdiff_t;

11.3.45 stdio.h

#define __IO_FILE_SIZE 148

extern char *const __sys_errlist(void);
extern void clearerr(FILE *);
extern int fclose(FILE *);
extern FILE *fopen(const char *, const char *);
extern int fprintf(FILE *, const char *, ...);
extern int fputc(int, FILE *);
extern FILE *freopen(const char *, const char *, FILE *);
extern FILE *fopen64(const char *, const char *, FILE *);
extern int fscanf(FILE *, const char *, ...);
extern int fseek(FILE *, long int, int);
extern int fseeko(FILE *, off_t, int);
extern int fscanf64(FILE *, const struct timespec *);
extern int ftello(FILE *);
extern int ftello64(FILE *);
extern int getc_unlocked(void);
extern int getw(FILE *);
extern int pclose(FILE *);
extern void perror(const char *);
extern FILE *popen(const char *, const char *);
extern int printf(const char *, ...);
extern int putc_unlocked(int, FILE *);
extern int putw(const char *, ..., FILE *);
extern int putc_unlocked(int, FILE *);
extern int putw(int, FILE *);
extern int remove(const char *);
extern void rewind(FILE *);
extern int scanf(const char *, ...);
extern void setbuf(FILE *, char *);
extern int sprintf(char *, const char *, ...);
extern int sscanf(const char *, const char *, ...);
extern FILE *stderr(void);
extern FILE *stdin(void);
extern FILE *stdout(void);
extern char *tempnam(const char *, const char *);
extern FILE *tmpfile64(void);
extern FILE *tmpfile(void);
extern char *tmpnam(char *);
extern int vfprintf(FILE *, const char *, va_list);
extern int vprintf(const char *, va_list);
extern int feof(FILE *
extern int ferror(FILE *
extern int fflush(FILE *
extern int fgetc(FILE *
extern int fgetpos(FILE *, fpos_t *
extern char *fgets(char *, int, FILE *
extern int fputs(const char *, FILE *
extern size_t fread(void *, size_t, size_t, FILE *
extern int fsetpos(FILE *, const fpos_t *
extern long int ftell(FILE *
extern size_t fwrite(const void *, size_t, size_t, FILE *
extern int getc(FILE *
extern int puts(const char *
extern int puts(const char *
extern int setvbuf(FILE *, char *, int, size_t);
extern int sprintf(char *, const char *, ...);
extern int ungetc(int, FILE *
extern int vsprintf(char *, size_t, const char *, va_list);
extern void flockfile(FILE *
extern int asprintf(char **, const char *, ...);
extern int fgetpos64(FILE *, fpos64_t *
extern FILE *fopen64(const char *, const char *);
extern int fsetpos64(FILE *, const fpos64_t *
extern int ftrylockfile(FILE *
extern void funlockfile(FILE *
extern int getc_unlocked(FILE *
extern void setbuffer(FILE *, char *, size_t);
extern int vasprintf(char **, const char *, va_list);
extern int vdprintf(int, const char *, va_list);
extern int vfscanf(FILE *, const char *, va_list);
extern int vscanf(const char *, va_list);
extern int vsscanf(const char *, const char *, va_list);
extern size_t __fpending(FILE *);

11.3.46 stdlib.h

extern double __strtod_internal(const char *, char **, int);
extern float __strtof_internal(const char *, char **, int);
extern long int __strtol_internal(const char *, char **, int, int);
extern long double __strtold_internal(const char *, char **, int);
extern long long int __strtoll_internal(const char *, char **, int, int);
extern unsigned long int __strtoull_internal(const char *, char **, int, int);
extern unsigned long long int __strtoull_internal(const char *, char **, int, int);
extern long int a64l(const char *);
extern void abort(void);
extern int abs(int);
extern double atof(const char *);
extern int atoi(char *);
extern long int atol(char *);
extern long long int atoll(const char *);
extern void *bsearch(const void *, const void *, size_t, size_t, __compar_fn_t);
extern div_t div(int, int);
extern double drand48(void);
extern char *ecvt(double, int, int *, int *);
extern double erand48(unsigned short);
extern void exit(int);
extern char *fcvt(double, int, char *, int *);
extern char *getenv(const char *);
extern int getsubopt(char **, char **, char **);
extern int grantpt(int);
extern long int jrand48(void);
extern long int labs(long int);
extern void lcong48(unsigned short);
extern ldiv_t ldiv(long int, long int);
extern long int llabs(long long int);
extern lldiv_t lldiv(long long int, long long int);
extern long int lrand48(void);
extern long int labs(long int);
extern size_t labs(long long int);
extern int lcong48(unsigned short);
extern long int ldiv(long int, long int);
extern long int llabs(long long int);
extern long int llabs(long long int);
extern long int lrand48(void);
extern long int mblen(const char *, size_t);
extern size_t mbstowcs(wchar_t *, const char *, size_t);
extern int mbtowc(wchar_t *, const char *, size_t);
extern char *mktemp(char *);
extern long int mrand48(void);
extern long int mrand48(void);
extern long int nrand48(unsigned short);
extern char *ptsname(int);
extern int putenv(char *);
extern void qsort(void *, size_t, size_t, __compar_fn_t);
extern int rand(void);
extern int rand_r(unsigned int *);
extern unsigned int *seed48(unsigned short);
extern int srand48(long int);
extern int unlockpt(int);
extern size_t wcstombs(char *, const wchar_t *, size_t);
extern int wcscmp(char *, wchar_t *);
extern int wcscoll(char *, wchar_t *);
extern int wcslen(const wchar_t *);
extern wchar_t *wcsnrtombs(const char *, const wchar_t *, size_t);
extern int wcsrtomb(char *, const wchar_t *, size_t);
extern int wcsrtombs(char *, const wchar_t *, size_t);
extern int wcsrtowc(char *, const wchar_t *, wchar_t *);
extern void *realpath(const char *, char *);
extern void realloc(void *, size_t);
extern char *setstate(char *);
extern void srand(unsigned int);
extern void srandom(unsigned int);
extern double strtod(char *, char **);
extern float strtof(const char *, char **);
extern long int strtol(char *, char **, int);
extern long double strtdold(const char *, char **);
extern long int strtof(const char *, char **, int);
extern long int strtoll(const char *, char **, int);
extern unsigned long int strtoul(const char *, char **, int);
extern unsigned long long int strtoull(const char *, char **, int);
extern unsigned long long int strtoull(const char *, char **, int);
extern unsigned long long int strtoull(const char *, char **, int);
extern void _Exit(int);
extern size_t __ctype_get_mb_cur_max(void);
extern char **environ(void);
extern char *realpath(const char *, char *);
11 Libraries

extern int mkstemp64(char *);
extern int posix_memalign(void **, size_t, size_t);
extern int posix_openpt(int);

11.3.47 string.h

extern void *__mempcpy(void *, const void *, size_t);
extern char *__stpcpy(char *, const char *);
extern char *__strtok_r(char *, const char *, char **);
extern void bcopy(void *, void *, size_t);
extern void *memchr(void *, int, size_t);
extern int memcmp(void *, void *, size_t);
extern void *memcpy(void *, void *, size_t);
extern void *memmem(const void *, size_t, const void *, size_t);
extern void *memmove(void *, const void *, size_t);
extern void *memset(void *, int, size_t);
extern char *strcat(char *, const char *);
extern char *strcmp(char *, char *);
extern int strcoll(const char *, const char *);
extern char *strcpy(char *, char *);
extern size_t strcspn(const char *, const char *);
extern char *_strerror(int);
extern size_t strlen(const char *);
extern char *strerror_r(int, char *, size_t);
extern void *memrchr(const void *, int, size_t);
extern char *rindex(char *, int);
extern int bcmp(void *, void *, size_t);
extern void bzer0(void *, size_t);
extern int ffs(int);
extern char *index(char *, int);
extern void *memccpy(void *, const void *, int, size_t);
extern char *index(char *, int);
extern int strcasecmp(char *, char *);
extern char *index(char *, int);
extern int strdup(char *);
extern int strndup(char *, size_t);
extern int strncasecmp(char *, char *, size_t);
extern int strndup(const char *, size_t);
extern int strncasecmp(const char *, char *, size_t);
extern int strlen(const char *);
extern int strsep(char **, const char *);
extern int strcasestr(const char *, const char *);
extern int stpcpy(char *, const char *);
extern int stpcpy(const void *, const char *, size_t);
extern void *memrchr(const void *, int, size_t);

11.3.48 sys/file.h

extern int flock(int, int);

11.3.49 sys/ioctl.h

#define TIOCGWINSZ 0x5413
11 Libraries

```c
#define FIONREAD        0x541B
#define TIOCNOTTY       0x5422

extern int ioctl(int, unsigned long int, ...);

11.3.50 sys/ipc.h

struct ipc_perm {
    key_t __key;
    uid_t uid;
    gid_t gid;
    uid_t cuid;
    gid_t cgid;
    unsigned short mode;
    unsigned short __pad1;
    unsigned short __seq;
    unsigned short __pad2;
    unsigned long int __unused1;
    unsigned long int __unused2;
};

extern key_t ftok(char *, int);

11.3.51 sys/mman.h

#define MCL_CURRENT     1
#define MCL_FUTURE      2

extern int msync(void *, size_t, int);
extern int mlock(const void *, size_t);
extern int mlockall(int);
extern void *mmap(void *, size_t, int, int, int, off_t);
extern int mprotect(void *, size_t, int);
extern int munlock(const void *, size_t);
extern int munlockall(void);
extern int munmap(void *, size_t);
extern void *mmap64(void *, size_t, int, int, int, off64_t);
extern int shm_open(const char *, int, mode_t);
extern int shm_unlink(const char *);

11.3.52 sys/msg.h

typedef unsigned long int msgqnum_t;
typedef unsigned long int msglen_t;

struct msqid_ds {
    struct ipc_perm msg_perm;
    time_t msg_stime;
    unsigned long int __unused1;
    time_t msg_rtime;
    unsigned long int __unused2;
    time_t msg_ctime;
    unsigned long int __unused3;
    unsigned long int __msg_cbytes;
    msgqnum_t msg_qnum;
    msglen_t msg_qbytes;
    pid_t msg_lspid;
    pid_t msg_lspid;
    unsigned long int __unused4;
    unsigned long int __unused5;
};
```
11 Libraries

extern int msgctl(int, int, struct msqid_ds *);
extern int msgget(key_t, int);
extern int msgrcv(int, void *, size_t, long int, int);
extern int msgsnd(int, const void *, size_t, int);

11.3.53 sys/param.h

/*
* This header is architecture neutral
* Please refer to the generic specification for details
*/

11.3.54 sys/poll.h

/*
* This header is architecture neutral
* Please refer to the generic specification for details
*/

11.3.55 sys/resource.h

extern int getpriority(__priority_which_t, id_t);
extern int getrlimit64(id_t, struct rlimit64 *);
extern int setpriority(__priority_which_t, id_t, int);
extern int setrlimit(__rlimit_resource_t, const struct rlimit *);
extern int setrlimit64(__rlimit_resource_t, const struct rlimit64 *);
extern int getrlimit(__rlimit_resource_t, struct rlimit *);
extern int getrusage(int, struct rusage *);

11.3.56 sys/sem.h

struct semid_ds {
    struct ipc_perm sem_perm;
    time_t sem_otime;
    unsigned long int __unused1;
    time_t sem_ctime;
    unsigned long int __unused2;
    unsigned long int __unused3;
    unsigned long int __unused4;
};
extern int semctl(int, int, int, ...);
extern int semget(key_t, int, int);
extern int semop(int, struct sembuf *, size_t);

11.3.57 sys/shm.h

#define SHMLBA (__getpagesize())
typedef unsigned long int shmat_t;

struct shmid_ds {
    struct ipc_perm shm_perm;
    int shm_segsz;
    time_t shm_atime;
    unsigned long int __unused1;
    time_t shm_dtime;
}
11 Libraries

unsigned long int __unused2;

time_t shm_ctime;

unsigned long int __unused3;

pid_t shm_cpid;

pid_t shm_lpid;

shmat_t shm_nattch;

unsigned long int __unused4;

unsigned long int __unused5;
};

extern int __getpagesize(void);
extern void *shmat(int, const void *, int);
extern int shmdt(const void *);
extern int shmget(key_t, size_t, int);

11.3.58 sys/socket.h

typedef uint32_t __ss_align_type;

#define SO_RCVLOWAT 18
#define SO_SNDLOWAT 19
#define SO_RCVTIMEO 20
#define SO_SNDFTIMEO 21

eextern int bind(int, const struct sockaddr *, socklen_t);
eextern int getnameinfo(const struct sockaddr *, socklen_t, char *,
socklen_t, char *, socklen_t, unsigned int);
eextern int listen(int, int);
eextern int setsockopt(int, int, int, const void *, socklen_t);
eextern int accept(int, struct sockaddr *, socklen_t *);
eextern int connect(int, const struct sockaddr *, socklen_t);
eextern ssize_t recv(int, void *, size_t, int);
eextern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *, socklen_t *);
eextern ssize_t recvmsg(int, struct msghdr *, int);
eextern ssize_t send(int, const void *, size_t, int,
const struct sockaddr *, socklen_t);
eextern int getpeername(int, struct sockaddr *, socklen_t *);
eextern int getsockopt(int, int, void *, socklen_t *);
eextern int shutdown(int, int);
eextern int socket(int, int, int);
eextern int socketpair(int, int, int);
eextern int sockatmark(int);

11.3.59 sys/stat.h

#define _STAT_VER 3

struct stat {
    dev_t st_dev;
    unsigned short __pad1;
    unsigned long int st_ino;
    mode_t st_mode;
    nlink_t st_nlink;
    pid_t st_uid;
    gid_t st_gid;
    dev_t st_rdev;
    unsigned short __pad2;
    off_t st_size;
};
11 Libraries

```c
blksize_t st_blksize;
blkcnt_t st_blocks;
struct timespec st_atim;
struct timespec st_mtim;
struct timespec st_ctim;
unsigned long int __unused4;
unsigned long int __unused5;
};

struct stat64 {
    dev_t st_dev;
    unsigned int __pad1;
    ino_t __st_ino;
    mode_t st_mode;
    nlink_t st_nlink;
    uid_t st_uid;
    gid_t st_gid;
    dev_t st_rdev;
    unsigned int __pad2;
    off64_t st_size;
    blksize_t st_blksize;
    blkcnt64_t st_blocks;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    ino64_t st_ino;
};

extern int __fxstat(int, int, struct stat *);
extern int __fxstat64(int, int, struct stat64 *);
extern int __lxstat(int, char *, struct stat *
); extern int __lxstat64(int, const char *, struct stat64 *
); extern int __xstat(int, const char *, struct stat *
); extern int __xstat64(int, const char *, struct stat64 *
); extern int mkfifo(const char *, mode_t); extern int chmod(const char *, mode_t);

11.3.60 sys/statvfs.h

struct statvfs {
    unsigned long int f_bsize;
    unsigned long int f_frsize;
    fsblkcant_t f_blocks;
    fsblkcant_t f_bfree;
    fsblkcant_t f_bavail;
    fsfilcnt_t f_files;
    fsfilcnt_t f_ffree;
    fsfilcnt_t f_favail;
    unsigned long int f_fsid;
    int __f_unused;
    unsigned long int f_flag;
    unsigned long int f_namemax;
    int __f_spare[6];
};

struct statvfs64 {
    unsigned long int f_bsize;
    unsigned long int f_frsize;
    fsblkcant64_t f_blocks;
    fsblkcant64_t f_bfree;
    fsblkcant64_t f_bavail;
    fsfilcnt64_t f_files;
```
11 Libraries

```c
  fsfilcnt64_t f_ffree;
  fsfilcnt64_t f_favail;
  unsigned long int f_fsid;
  int __f_unused;
  unsigned long int f_flag;
  unsigned long int f_namemax;
  int __f_spare[6];
};

extern int fstatvfs(int, struct statvfs *);
extern int fstatvfs64(int, struct statvfs64 *);
extern int statvfs(const char *, struct statvfs *);
extern int statvfs64(const char *, struct statvfs64 *);

11.3.61 sys/time.h

extern int getitimer(__itimer_which_t, struct itimerval *);
extern int setitimer(__itimer_which_t, const struct itimerval *,
                      struct itimerval *);
extern int adjtime(const struct timeval *, struct timeval *);
extern int gettimeofday(struct timeval *, struct timezone *);
extern int utimes(const char *, const struct timeval *);

11.3.62 sys/timeb.h

extern int ftime(struct timeb *);

11.3.63 sys/times.h

extern clock_t times(struct tms *);

11.3.64 sys/types.h

typedef long long int int64_t;
typedef int32_t ssize_t;
#define __FDSET_LONGS 32

11.3.65 sys/uio.h

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

11.3.66 sys/un.h

/**
 * This header is architecture neutral
 * Please refer to the generic specification for details
 */

11.3.67 sys/utsname.h

extern int uname(struct utsname *);
```
11 Libraries

11.3.68 sys/wait.h

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

11.3.69 syslog.h

extern void closelog(void);
extern void openlog(const char *, int, int);
extern int setlogmask(int);
extern void syslog(int, const char *, ...);
extern void vsyslog(int, const char *, va_list);

11.3.70 termios.h

#define OLCUC   0000002
#define ONLCR   0000004
#define XCASE   0000004
#define NLDLY   0000400
#define CR1     0001000
#define IUCLC   0001000
#define CR2     0002000
#define CR3     0003000
#define CRDLY   0003000
#define TAB1    0004000
#define TAB2    0010000
#define TAB3    0014000
#define TABDLY  0014000
#define BS1     0020000
#define BSDLY   0020000
#define VT1     0040000
#define VTDLY   0040000
#define FF1     0100000
#define FFDLY   0100000
#define VSUSP   10
#define VEOL    11
#define VREPRINT        12
#define VDISCARD        13
#define VWERASE 14
#define VEOLO 16
#define VMIN   6
#define VSWTC  7
#define VSTART 8
#define VSTOP   9
#define IXON    0002000
#define IXOFF   0010000
#define CS6     0000020
#define CS7     0000040
#define CS8     0000060
#define CSIZE   0000060
#define CSTOPB  0000100
#define CREAD   0000200
#define PARENB  0000400
#define PARODD  0001000
#define HUPCL   0002000
#define CLOCAL  0004000
#define VTIME   5
#define ISIG    0000001
#define ICANON  0000002
#define ECHOE   0000020
#define ECHOK    0000040
#define ECHONL   0000100
#define NOFLSH   0000200
#define TOSTOP   0000400
#define ECHOCTL  0001000
#define ECHOPRT  0002000
#define ECHOKE   0004000
#define FLUSHO   0010000
#define PENDIN   0040000
#define IEXTEN   0100000

extern speed_t cfgetispeed(const struct termios *);
extern speed_t cfgetospeed(const struct termios *);
extern void cfmakeraw(struct termios *);
extern int cfsetispeed(struct termios *, speed_t);
extern int cfsetospeed(struct termios *, speed_t);
extern int cfsetspeed(struct termios *, speed_t);
extern int tcflow(int, int);
extern int tcflush(int, int);
extern pid_t tcgetsid(int);
extern int tcsendbreak(int, int);
extern int tcsetattr(int, int, const struct termios *);
extern int tcdrain(int);
extern int tcgetattr(int, struct termios *);

11.3.71 time.h

extern int __daylight(void);
extern long int __timezone(void);
extern char *__tzname(void);
extern char *asctime(const struct tm *);
extern clock_t clock(void);
extern char *ctime(const time_t *);
extern char *ctime_r(const time_t *, char *);
extern double difftime(time_t, time_t);
extern struct tm *getdate(const char *);
extern int getdate_err(void);
extern struct tm *gmtime(const time_t *);
extern struct tm *localtime(const time_t *);
extern time_t mktime(struct tm *);
extern int stime(const time_t *);
extern size_t strftime(char *, size_t, const char *, const struct tm *);
extern char *strptime(const char *, const char *, struct tm *);
extern time_t time(time_t *);
extern int nanosleep(const struct timespec *, struct timespec *);
11 Libraries

extern int timer_create(clockid_t, struct sigevent *, timer_t *);
extern int timer_delete(timer_t);
extern int timer_getoverrun(timer_t);
extern int timer_gettime(timer_t, struct itimerspec *);
extern int timer_settime(timer_t, int, const struct itimerspec *,
                        struct itimerspec *);

11.3.72 ucontext.h

typedef int greg_t;

#define NGREG 19
typedef greg_t gregset_t[19];

struct _libc_fpreg {
    unsigned short significand[4];
    unsigned short exponent;
};

struct _libc_fpstate {
    unsigned long int cw;
    unsigned long int sw;
    unsigned long int tag;
    unsigned long int ipoff;
    unsigned long int cssel;
    unsigned long int dataoff;
    unsigned long int dataset;
    struct _libc_fpreg_st[8];
    unsigned long int status;
};
typedef struct _libc_fpstate *fpregset_t;

typedef struct {
    gregset_t gregs;
    fpregset_t fpregs;
    unsigned long int oldmask;
    unsigned long int cr2;
} mcontext_t;

typedef struct ucontext {
    unsigned long int uc_flags;
    struct ucontext *uc_link;
    stack_t uc_stack;
    mcontext_t uc_mcontext;
    sigset_t uc_sigmask;
    struct _libc fpstate __fpregs_mem;
} ucontext_t;
extern int getcontext(ucontext_t *);
extern int makecontext(ucontext_t *, void (*func) (void)
                        , int, ...);
extern int setcontext(const struct ucontext *);
extern int swapcontext(ucontext_t *, const struct ucontext *);

11.3.73 ulimit.h

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

11.3.74 unistd.h

extern long int ulimit(int, ...);
typedef int intptr_t;

extern char **__environ(void);
extern pid_t __getpgid(pid_t);
extern void __exit(int);
extern int acct(const char *);
extern unsigned int alarm(unsigned int);
extern int chown(const char *, uid_t, gid_t);
extern int chroot(const char *);
extern size_t confstr(int, char *, size_t);
extern int creat(const char *, mode_t);
extern int creat64(const char *, mode_t);
extern char *ctermid(char *);
extern char *cuserid(char *);
extern int daemon(int, int);
extern int execl(const char *, const char *, ...);
extern int execle(const char *, const char *, ...);
extern int execlp(const char *, const char *, ...);
extern int execv(const char *, char *const);
extern int execvp(const char *, char *const);
extern int fdatasync(int);
extern int ftruncate64(int, off64_t);
extern long int gethostid(void);
extern char *getlogin(void);
extern int getlogin_r(char *, size_t);
extern int getopt(int, char *const, const char *);
extern pid_t getpgrp(void);
extern pid_t getsid(pid_t);
extern char *getwd(char *);
extern int lockf(int, int, off_t);
extern int mkstemp(char *);
extern int nice(int);
extern char *optarg(void);
extern int opterr(void);
extern int optind(void);
extern int optopt(void);
extern int rename(const char *, const char *);
extern int setegid(gid_t);
extern int seteuid(uid_t);
extern int sethostname(const char *, size_t);
extern int setpgrp(void);
extern int swab(const void *, void *, ssize_t);
extern int sync(void);
extern pid_t tcgetpgrp(int);
extern int tcsetpgrp(int, pid_t);
extern int truncate(const char *, off_t);
extern int truncate64(const char *, off64_t);
extern char *ttyname(int);
extern unsigned int ualarm(useconds_t, useconds_t);
extern int usleep(useconds_t);
extern int close(int);
extern int fsync(int);
extern int lseek(int, off_t, int);
extern int open(const char *, int, ...);
extern int pause(void);
extern ssize_t read(int, void *, size_t);
extern ssize_t write(int, const void *, size_t);
extern char *crypt(char *, char *);
extern void encrypt(char *, int);
extern void setkey(const char *);
extern int access(const char *, int);
extern int brk(void *);
extern int chdir(const char *);
11 Libraries

1305 extern int dup2(int, int);
1306 extern int execve(const char *, char *const, char *const);
1307 extern int fchdir(int);
1308 extern int gid_t fork(void);
1309 extern int getegid(void);
1310 extern int getgid(void);
1311 extern int getgroups(int, gid_t);
1312 extern int gethostname(char *, size_t);
1313 extern int getpgid(pid_t);
1314 extern int getpid(void);
1315 extern int getuid(void);
1316 extern int lchown(const char *, uid_t, gid_t);
1317 extern int link(const char *, const char *);
1318 extern int mkdir(const char *, mode_t);
1319 extern long int pathconf(const char *, int);
1320 extern int pipe(int);
1321 extern int readlink(const char *, char *, size_t);
1322 extern void *sbrk(ptrdiff_t);
1323 extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
1324 extern int setgid(gid_t);
1325 extern int setpgid(pid_t, pid_t);
1326 extern int setregid(gid_t, gid_t);
1327 extern int setreuid(uid_t, uid_t);
1328 extern int setsid(void);
1329 extern int setuid(uid_t);
1330 extern int sleep(unsigned int);
1331 extern int symlink(const char *, const char *);
1332 extern int unlink(const char *);
1333 extern pid_t vfork(void);
1334 extern ssize_t pread64(int, void *, size_t, off64_t);
1335 extern ssize_t pwrite64(int, const void *, size_t, off64_t);
1336 extern ssize_t pread(int, void *, size_t, off_t);
1337 extern ssize_t pwrite(int, const void *, size_t, off_t);
1338 extern char **_environ(void);
1339 extern long int fpathconf(int, int);
1340 extern pid_t setuid(uid_t);
1341 extern unsigned int sleep(unsigned int);
1342 extern int symlink(const char *, const char *);
1343 extern long int sysconf(int);
1344 extern int unlink(const char *);
1345 extern int unlink(const char *);
1346 extern int vfork(void);
1347 extern ssize_t pread64(int, void *, size_t, off_t);
1348 extern ssize_t pwrite64(int, const void *, size_t, off64_t);
1349 extern char **_environ(void);
1350 extern int utime(const char *, const struct utimbuf *);
1351 extern int utime(const char *, const struct utimbuf *);
1352 11.3.75 utime.h

1353 struct lastlog {
1354     time_t ll_time;
1355     char ll_line[UT_LINESIZE];
1356     char ll_host[UT_HOSTSIZE];
1357 }
1358
1359 struct utmp {
1360     short ut_type;
11.3.77 utmpx.h

```c
struct utmpx {
    short ut_type;
    pid_t ut_pid;
    char ut_line[UT_LINESIZE];
    char ut_id[4];
    char ut_user[UT_NAMESIZE];
    char ut_host[UT_HOSTSIZE];
    struct exit_status ut_exit;
    long int ut_session;
    struct timeval ut_tv;
    int32_t ut_addr_v6[4];
    char __unused[20];
};
```

11.3.78 wchar.h

```c
extern double __wcstod_internal(const wchar_t *, wchar_t **, int);
extern float __wcstof_internal(const wchar_t *, wchar_t **, int);
extern long int __wcstol_internal(const wchar_t *, wchar_t **, int, int);
extern long double __wcstold_internal(const wchar_t *, wchar_t **, int);
extern unsigned long int __wcstoul_internal(const wchar_t *, wchar_t **,
                                            int, int);
extern wchar_t *wcscat(wchar_t *, const wchar_t *);
extern wchar_t *wcschr(const wchar_t *, wchar_t *);
extern int wcscmp(const wchar_t *, const wchar_t *);
extern int wcscoll(const wchar_t *, const wchar_t *);
extern wchar_t *wcsncpy(const wchar_t *, const wchar_t *);
extern wchar_t *wcsrev(const wchar_t *);
extern size_t wcsxspn(const wchar_t *, const wchar_t *);
```
extern wchar_t *wcsncat(wchar_t *, const wchar_t *, size_t);
extern int wcsncmp(const wchar_t *, const wchar_t *, size_t);
extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
extern size_t wcsspn(const wchar_t *, const wchar_t *);
extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t * *);
extern int wcswidth(const wchar_t *, size_t);
extern size_t wcssxfrm(wchar_t *, const wchar_t *, size_t);
extern int wcstob(wint_t);
extern int wcwidth(wchar_t);
extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
extern size_t mbrlen(const char *, size_t, mbstate_t *);
extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
extern int mbsinit(const mbstate_t *);
extern size_t mbsnrtows(wchar_t *, const char **, size_t, mbstate_t *);
extern size_t wcsrtombs(char *, const wchar_t **, size_t, mbstate_t *);
extern wchar_t *wcscpy(wchar_t *, const wchar_t *);
extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
extern size_t wcrtomb(char *, wchar_t, mbstate_t *);
extern size_t wcslen(const wchar_t *);
extern size_t wcsnrtombs(char *, const wchar_t *, size_t, size_t, mbstate_t *);
extern size_t wcsrtombs(char *, const wchar_t *, size_t, mbstate_t *);
extern double wcstod(const wchar_t *, wchar_t * * );
extern long int wcstol(const wchar_t *, wchar_t * *, int);
extern long double wcstold(const wchar_t *, wchar_t * *);
extern unsigned long int wcstoul(const wchar_t *, wchar_t * *, int);
extern unsigned long long int wcstoull(const wchar_t *, wchar_t * *, int);
extern wchar_t *wcscsncmp(const wchar_t *, const wchar_t *);
extern size_t wcscsncmp(const wchar_t *, const wchar_t *, size_t);
extern size_t wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
extern size_t wcsnlen(const wchar_t *, size_t);
extern long long int wcstoll(const wchar_t *, wchar_t * *, int);
extern unsigned long long int wcstoull(const wchar_t *, wchar_t * *, int);
extern wint_t btowc(int);
extern wint_t fgetwc(FILE *);
extern wint_t fgetwc_unlocked(FILE *);
extern wint_t *fgetws(wchar_t *, int, FILE *);
extern wint_t fputwc(const wchar_t *, FILE *);
extern int fputws(const wchar_t *, FILE *);
extern int fwprintf(FILE *, const wchar_t *, ...);
extern int fwscanf(const wchar_t *, const wchar_t *, ...);
extern wint_t getwc(FILE *);
extern int ungetwc(const wchar_t *, int);
extern wint_t getwchar(void);
extern wint_t putwc(const wchar_t *, FILE *);
extern wint_t putwchar(const wchar_t *);
extern wint_t swprintf(const wchar_t *, size_t, const wchar_t *, ...);
extern size_t swscanf(const wchar_t *, const wchar_t *, ...);
extern wint_t ungetwc(const wchar_t *, FILE *);
extern int vfwprintf(FILE *, const wchar_t *, va_list);
extern int vfwscanf(FILE *, const wchar_t *, va_list);
extern int vsprintf(const wchar_t *, size_t, const wchar_t *, va_list);
extern int vsscanf(const wchar_t *, const wchar_t *, va_list);
extern int vswprintf(const wchar_t *, va_list);
extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
11 Libraries

extern int vwscanf(const wchar_t *, va_list);
extern size_t wcsftime(wchar_t *, size_t, const wchar_t *,
    const struct tm *);
extern int wprintf(const wchar_t *, ...);
extern int wscanf(const wchar_t *, ...);

11.3.79 wctype.h

extern int iswblank(wint_t);
extern wint_t towlower(wint_t);
extern wint_t towupper(wint_t);
extern wctrans_t wctrans(const char *);
extern int iswalnum(wint_t);
extern int iswalpha(wint_t);
extern int iswcntrl(wint_t);
extern int iswctype(wint_t, wctype_t);
extern int iswdigit(wint_t);
extern int iswgraph(wint_t);
extern int iswlower(wint_t);
extern int iswprint(wint_t);
extern int iswpunct(wint_t);
extern int iswspace(wint_t);
extern int iswupper(wint_t);
extern int iswxdigit(wint_t);
extern wctype_t wctype(const char *);
extern wint_t towctrans(wint_t, wctrans_t);

11.3.80 wordexp.h

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

11.4 Interfaces for libm

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

Table 11-24 libm Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libm</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libm.so.6</td>
</tr>
</tbody>
</table>

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

[LSB] This Specification
[SUSv2] SUSv2

11.4.1 Math

11.4.1.1 Interfaces for Math

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.
### Table 11-25 libm - Math Function Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>__finite(GLIBC_2.1) [ISOC99]</th>
<th>__finitef(GLIBC_2.0) [ISOC99]</th>
<th>__finitel(GLIBC_2.1) [ISOC99]</th>
<th>__fpclassify(GLIBC_2.1) [LSB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>__finite(GLIBC_2.0)</td>
<td>__finite(GLIBC_2.0) [ISOC99]</td>
<td>__finitef(GLIBC_2.0) [ISOC99]</td>
<td>__finitel(GLIBC_2.0) [ISOC99]</td>
<td>__fpclassify(GLIBC_2.0) [LSB]</td>
</tr>
<tr>
<td>__finitef(GLIBC_2.0)</td>
<td>__finitef(GLIBC_2.0) [ISOC99]</td>
<td>__finitef(GLIBC_2.0) [ISOC99]</td>
<td>__finitef(GLIBC_2.0) [ISOC99]</td>
<td>__finitef(GLIBC_2.0) [ISOC99]</td>
</tr>
<tr>
<td>__finitel(GLIBC_2.0)</td>
<td>__finitel(GLIBC_2.0) [ISOC99]</td>
<td>__finitel(GLIBC_2.0) [ISOC99]</td>
<td>__finitel(GLIBC_2.0) [ISOC99]</td>
<td>__finitel(GLIBC_2.0) [ISOC99]</td>
</tr>
<tr>
<td>__fpclassify(GLIBC_2.0)</td>
<td>__fpclassify(GLIBC_2.0) [LSB]</td>
<td>__fpclassify(GLIBC_2.0) [LSB]</td>
<td>__fpclassify(GLIBC_2.0) [LSB]</td>
<td>__fpclassify(GLIBC_2.0) [LSB]</td>
</tr>
<tr>
<td>_finite(GLIBC_2.0) [SUSv3]</td>
<td>_finite(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_finite(GLIBC_2.0) [SUSv3]</td>
<td>_finite(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
<td>_finitef(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
<td>_finitel(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
<td>_fpclassify(GLIBC_2.0) [SUSv3]</td>
</tr>
</tbody>
</table>

### Example Functions

- **acos(GLIBC_2.0)** [SUSv3]
- **acosf(GLIBC_2.0)** [SUSv3]
- **acosh(GLIBC_2.0)** [SUSv3]
- **acoshf(GLIBC_2.0)** [SUSv3]
- **acosl(GLIBC_2.0)** [SUSv3]
- **asin(GLIBC_2.0)** [SUSv3]
- **asinf(GLIBC_2.0)** [SUSv3]
- **asin(GLIBC_2.0)** [SUSv3]
- **asinh(GLIBC_2.0)** [SUSv3]
- **asinhf(GLIBC_2.0)** [SUSv3]
- **asinhl(GLIBC_2.0)** [SUSv3]
- **asinl(GLIBC_2.0)** [SUSv3]
- **atan(GLIBC_2.0)** [SUSv3]
- **atan2(GLIBC_2.0)** [SUSv3]
- **atan2f(GLIBC_2.0)** [SUSv3]
- **atan2l(GLIBC_2.0)** [SUSv3]
- **atanf(GLIBC_2.0)** [SUSv3]
- **atanh(GLIBC_2.0)** [SUSv3]
- **atanhf(GLIBC_2.0)** [SUSv3]
- **atanhl(GLIBC_2.0)** [SUSv3]
- **atanl(GLIBC_2.0)** [SUSv3]
- **cabs(GLIBC_2.1)** [SUSv3]
- **cabsf(GLIBC_2.1)** [SUSv3]
- **cabsl(GLIBC_2.1)** [SUSv3]
- **cacos(GLIBC_2.1)** [SUSv3]
- **cacosf(GLIBC_2.1)** [SUSv3]
- **cacosh(GLIBC_2.1)** [SUSv3]
- **cacoshf(GLIBC_2.1)** [SUSv3]
- **cacoshl(GLIBC_2.1)** [SUSv3]
- **cacosl(GLIBC_2.1)** [SUSv3]
- **carg(GLIBC_2.1)** [SUSv3]
- **cargf(GLIBC_2.1)** [SUSv3]
- **cargl(GLIBC_2.1)** [SUSv3]
- **casin(GLIBC_2.1)** [SUSv3]
- **casinf(GLIBC_2.1)** [SUSv3]
- **casinh(GLIBC_2.1)** [SUSv3]
- **casinhf(GLIBC_2.1)** [SUSv3]
- **casinhl(GLIBC_2.1)** [SUSv3]
- **casinl(GLIBC_2.1)** [SUSv3]
- **catan(GLIBC_2.1)** [SUSv3]
- **catanf(GLIBC_2.1)** [SUSv3]
- **catanh(GLIBC_2.1)** [SUSv3]
- **catanhf(GLIBC_2.1)** [SUSv3]
- **catanhf(GLIBC_2.1)** [SUSv3]
- **catanhl(GLIBC_2.1)** [SUSv3]
- **catanl(GLIBC_2.1)** [SUSv3]
- **cbrt(GLIBC_2.0)** [SUSv3]
- **cbrtf(GLIBC_2.0)** [SUSv3]
- **cbrtl(GLIBC_2.0)** [SUSv3]
- **ccos(GLIBC_2.1)** [SUSv3]
- **ccosf(GLIBC_2.1)** [SUSv3]
- **ccosh(GLIBC_2.1)** [SUSv3]
- **ccoshf(GLIBC_2.1)** [SUSv3]
- **ccoshl(GLIBC_2.1)** [SUSv3]
- **ccosl(GLIBC_2.1)** [SUSv3]
- **ceil(GLIBC_2.0)** [SUSv3]
- **ceilf(GLIBC_2.0)** [SUSv3]
- **ceill(GLIBC_2.0)** [SUSv3]
- **cexp(GLIBC_2.1)** [SUSv3]
- **cexpf(GLIBC_2.1)** [SUSv3]
- **cexpl(GLIBC_2.1)** [SUSv3]
- **cimag(GLIBC_2.1)** [SUSv3]
- **cimagf(GLIBC_2.1)** [SUSv3]
- **cimagl(GLIBC_2.1)** [SUSv3]
- **clog(GLIBC_2.1)** [SUSv3]
- **clog10(GLIBC_2.1)** [ISOC99]
- **clog10f(GLIBC_2.1)** [ISOC99]
- **clog10l(GLIBC_2.1)** [ISOC99]
- **clogf(GLIBC_2.1)** [SUSv3]
- **clogf(GLIBC_2.1)** [SUSv3]
- **conj(GLIBC_2.1)** [SUSv3]
- **conjf(GLIBC_2.1)** [SUSv3]
- **conjf(GLIBC_2.1)** [SUSv3]
<table>
<thead>
<tr>
<th>Function</th>
<th>Library</th>
<th>Function</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>conjl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>copysign(GLIBC_2.0) [SUSv3]</td>
<td>copysign(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>cos(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>cosh(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>cosh(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>cpow(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>cproj(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>cprojf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>cprojl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>cprojl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>creal(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>crealf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>creall(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csin(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>csinf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csinh(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>csinhf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csinhl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>csinl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csqrt(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>csqrtf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csqrtl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>csinl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csqrtf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>csinl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>csqrtl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ctan(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>ctanf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ctanh(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>ctanhf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>ctanl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>ctanl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>erf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>erfc(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>erfcf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>erfcпло(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>erfl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>exp(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>exp2(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>exp2f(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>exp2f(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>exp2l(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>expm1(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>expm1f(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>expm1l(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>fabs(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fabsf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>fabsl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fabsl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>fdim(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fdimf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>fdiml(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fclearexcept(GLIBC_2.2)</td>
<td>[SUSv3]</td>
<td>fegetenv(GLIBC_2.0) [SUSv3]</td>
<td>fegetenv(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>fgetenv(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td>fgetenv(GLIBC_2.0) [SUSv3]</td>
<td>fegetenv(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>fgetround(GLIBC_2.1) [SUSv3]</td>
<td></td>
<td>fegetround(GLIBC_2.1) [SUSv3]</td>
<td>fegetround(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>fetestexceptflag(GLIBC_2.2) [SUSv3]</td>
<td></td>
<td>fetestexceptflag(GLIBC_2.2) [SUSv3]</td>
<td>fetestexceptflag(GLIBC_2.2) [SUSv3]</td>
</tr>
<tr>
<td>finite(GLIBC_2.0)</td>
<td>[SUSv2]</td>
<td>fminf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmax(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>fmin(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmaxf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>fminl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>fmaxl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td>fmod(GLIBC_2.0)</td>
<td>[SUSv3]</td>
</tr>
<tr>
<td>Function</td>
<td>Library</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------</td>
<td>-------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>fmodf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>frexpf(GLIBC_2.0)</td>
</tr>
<tr>
<td>fmodl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>frexp(GLIBC_2.0)</td>
</tr>
<tr>
<td>frexpl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>gammal(GLIBC_2.0)</td>
</tr>
<tr>
<td>hypotf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>hypotl(GLIBC_2.0)</td>
</tr>
<tr>
<td>ilogbf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>j0(GLIBC_2.0)</td>
</tr>
<tr>
<td>ilogbl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>j1(GLIBC_2.0)</td>
</tr>
<tr>
<td>j0(GLIBC_2.0)</td>
<td>[ISOC99]</td>
<td></td>
<td>j0f(GLIBC_2.0)</td>
</tr>
<tr>
<td>j1(GLIBC_2.0)</td>
<td>[ISOC99]</td>
<td></td>
<td>j1f(GLIBC_2.0)</td>
</tr>
<tr>
<td>jnf(GLIBC_2.0)</td>
<td>[ISOC99]</td>
<td></td>
<td>j1l(GLIBC_2.0)</td>
</tr>
<tr>
<td>jn(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>j0(GLIBC_2.0)</td>
</tr>
<tr>
<td>jnf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>j1f(GLIBC_2.0)</td>
</tr>
<tr>
<td>jnl(GLIBC_2.0)</td>
<td>[ISOC99]</td>
<td></td>
<td>j1l(GLIBC_2.0)</td>
</tr>
<tr>
<td>ldexpf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>ldexplf(GLIBC_2.0)</td>
</tr>
<tr>
<td>ldexpl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>lgammaf(GLIBC_2.0)</td>
</tr>
<tr>
<td>llrintf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>llrintl(GLIBC_2.1)</td>
</tr>
<tr>
<td>llrnf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>llrnl(GLIBC_2.0)</td>
</tr>
<tr>
<td>llrint(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>llrintf(GLIBC_2.1)</td>
</tr>
<tr>
<td>llrintl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>llrintl(GLIBC_2.1)</td>
</tr>
<tr>
<td>llroundf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>log(GLIBC_2.0)</td>
</tr>
<tr>
<td>llroundl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>log(GLIBC_2.0)</td>
</tr>
<tr>
<td>log10f(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>log10f(GLIBC_2.0)</td>
</tr>
<tr>
<td>log10l(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>log10l(GLIBC_2.0)</td>
</tr>
<tr>
<td>log1pf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>log1pf(GLIBC_2.0)</td>
</tr>
<tr>
<td>log2f(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>log2f(GLIBC_2.1)</td>
</tr>
<tr>
<td>log2l(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>log2l(GLIBC_2.1)</td>
</tr>
<tr>
<td>logf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>logf(GLIBC_2.0)</td>
</tr>
<tr>
<td>log1f(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>log1f(GLIBC_2.0)</td>
</tr>
<tr>
<td>log1l(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>log1l(GLIBC_2.0)</td>
</tr>
<tr>
<td>log1f(GLIBC_2.0)</td>
<td>[ISOC99]</td>
<td></td>
<td>log1f(GLIBC_2.0)</td>
</tr>
<tr>
<td>lrint(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>lrintf(GLIBC_2.1)</td>
</tr>
<tr>
<td>lrintl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>lrintf(GLIBC_2.1)</td>
</tr>
<tr>
<td>llroundf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>llroundf(GLIBC_2.1)</td>
</tr>
<tr>
<td>llroundl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>llroundl(GLIBC_2.1)</td>
</tr>
<tr>
<td>nannf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nannf(GLIBC_2.1)</td>
</tr>
<tr>
<td>nannf(GLIBC_2.1)</td>
<td>[ISOC99]</td>
<td></td>
<td>nannf(GLIBC_2.1)</td>
</tr>
<tr>
<td>nearbyintf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nearbyintf(GLIBC_2.1)</td>
</tr>
<tr>
<td>nearbyintf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nearbyintf(GLIBC_2.1)</td>
</tr>
<tr>
<td>nextafterf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>nextafterf(GLIBC_2.0)</td>
</tr>
<tr>
<td>nextafterf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>nextafterf(GLIBC_2.0)</td>
</tr>
<tr>
<td>nextafterl(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>nextafterl(GLIBC_2.0)</td>
</tr>
<tr>
<td>nearbyintl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nearbyintl(GLIBC_2.1)</td>
</tr>
<tr>
<td>nearbyintl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nearbyintl(GLIBC_2.1)</td>
</tr>
<tr>
<td>nexttowardf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nexttowardf(GLIBC_2.1)</td>
</tr>
<tr>
<td>nexttowardf(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nexttowardf(GLIBC_2.1)</td>
</tr>
<tr>
<td>nexttowardl(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>nexttowardl(GLIBC_2.1)</td>
</tr>
<tr>
<td>pow10f(GLIBC_2.1)</td>
<td>[ISOC99]</td>
<td></td>
<td>pow10f(GLIBC_2.1)</td>
</tr>
<tr>
<td>pow10f(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>pow10f(GLIBC_2.1)</td>
</tr>
<tr>
<td>powf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>powf(GLIBC_2.0)</td>
</tr>
<tr>
<td>pow10l(GLIBC_2.1)</td>
<td>[ISOC99]</td>
<td></td>
<td>pow10l(GLIBC_2.1)</td>
</tr>
<tr>
<td>pow10l(GLIBC_2.1)</td>
<td>[SUSv3]</td>
<td></td>
<td>pow10l(GLIBC_2.1)</td>
</tr>
<tr>
<td>powf(GLIBC_2.0)</td>
<td>[SUSv3]</td>
<td></td>
<td>powf(GLIBC_2.0)</td>
</tr>
</tbody>
</table>
### Table 11-26: libm - Math Data Interfaces

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>signgam(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>remainder(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>remainderf(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>remainderl(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>remquo(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>remquof(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>remquol(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>rint(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>rintl(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>round(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>roundf(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>roundl(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>scalb(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>scalbf(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>scalbl(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>scalbln(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>scalblnf(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>scalblnl(GLIBC_2.1)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>significand(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>significandf(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>significandl(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>sin(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sinh(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sinhf(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sinhl(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sinh(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sinh(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sqrt(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>sqrt(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>tan(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>tanh(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>tanh(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>y0(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>y0f(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>y0l(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>y1(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>y1f(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>y1l(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>yn(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>SUSv3</td>
</tr>
<tr>
<td>ynf(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
<tr>
<td>ynlf(GLIBC_2.0)</td>
<td>11 Libraries</td>
<td>ISOC99</td>
</tr>
</tbody>
</table>

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.

### 11.5 Data Definitions for libm

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

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

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

### 11.5.1 complex.h

```c
extern double cabs(double complex);
extern float cabsf(float complex);
extern long double cabsl(long double complex);
extern double complex cacos(double complex);
extern float complex cacosf(float complex);
extern long double complex cacosl(long double complex);
extern double complex cacosh(double complex);
extern float complex cacoshf(float complex);
extern long double complex cacoshl(long double complex);
extern double complex cacosl(long double complex);
extern double complex carg(double complex);
extern float cargf(float complex);
extern long double complex cargl(long double complex);
extern double complex casin(double complex);
extern float complex casinf(float complex);
extern long double complex casinh(double complex);
extern float complex casinhf(float complex);
extern double complex casinl(long double complex);
extern double complex casinhl(long double complex);
extern double complex casinl(long double complex);
extern double complex catan(double complex);
extern float complex catanf(float complex);
extern double complex catanh(double complex);
extern float complex catanhf(float complex);
extern long double complex catanhl(long double complex);
extern double complex catanl(long double complex);
extern double complex cconj(double complex);
extern float complex cconjr(float complex);
extern long double complex cconjl(long double complex);
extern double complex ccosh(double complex);
extern float complex ccoshf(float complex);
extern long double complex ccoshl(long double complex);
extern double complex ccosl(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexpl(long double complex);
extern double complex cimag(double complex);
extern float complex cimagf(float complex);
extern long double complex cimagl(long double complex);
extern double complex cimagj(long double complex);
extern double complex clog(double complex);
extern float complex clog10f(float complex);
extern long double complex log10l(long double complex);
extern double complex clog10l(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexp1l(long double complex);
extern double complex cimag(double complex);
extern float complex cimagf(float complex);
extern long double complex cimagl(long double complex);
extern double complex cimagj(long double complex);
extern double complex clog(double complex);
extern float complex clog10f(float complex);
extern long double complex clog10l(long double complex);
extern double complex clog10l(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexp1l(long double complex);
extern double complex cimag(double complex);
extern float complex cimagf(float complex);
```
extern long double complex cpowl(long double complex, long double complex);
extern double complex cproj(double complex);
extern float complex cprojf(float complex);
extern long double complex cprojl(long double complex);
extern float crealf(float complex);
extern double complex csinl(long double complex);
extern double complex cprojl(long double complex);
extern double complex csinl(long double complex);
extern double complex csqrtl(long double complex);
extern long double complex csinl(long double complex);
extern double complex ctanl(long double complex);
extern long double complex ctanl(long double complex);

11.5.2 fenv.h

#define FE_INVALID 0x01
#define FE_DIVBYZERO 0x04
#define FE_OVERFLOW 0x08
#define FE_UNDERFLOW 0x10
#define FE_INEXACT 0x20

#define FE_ALL_EXCEPT (FE_INEXACT | FE_DIVBYZERO | FE_UNDERFLOW | FE_OVERFLOW | FE_INVALID)

#define FE_TONEAREST 0
#define FE_DOWNWARD 0x400
#define FE_UPWARD 0x800
#define FE_TOWARDZERO 0xc00

typedef unsigned short fexcept_t;
typedef struct {
    unsigned short __control_word;
    unsigned short __unused1;
    unsigned short __status_word;
    unsigned short __unused2;
    unsigned short __tags;
    unsigned short __unused3;
    unsigned int __eip;
    unsigned short __cs_selector;
    unsigned int __opcode:11;
    unsigned int __unused4:5;
    unsigned int __data_offset;
    unsigned short __data_selector;
    unsigned short __unused5;
} fenv_t;

#define FE_DFL_ENV ((__const fenv_t *) -1)

extern int feclearexcept(int);
extern int fegetenv(fenv_t *);
11 Libraries

extern int fegetexceptflag(fexcept_t *f, int);
extern int fegetround(void);
extern int feholdexcept(fenv_t *e);
extern int feraiseexcept(int);
extern int fesetenv(const fenv_t *e);
extern int fesetexceptflag(const fexcept_t *f, int);
extern int fesetround(int);
extern int fetestexcept(int);
extern int feupdateenv(const fenv_t *e);

11.5.3 math.h

#define fpclassify(x) \  
(sizeof (x) == sizeof (float) ? __fpclassifyf (x) : sizeof (x) == sizeof (double) ? __fpclassify (x) : __fpclassifyl (x))
#define signbit(x) \  
(sizeof (x) == sizeof (float) ? __signbitf (x) : sizeof (x) == sizeof (double) ? __signbit (x) : __signbitl (x))

#define FP_ILOGB0 (-2147483647 - 1)
#define FP_ILOGBNAN (-2147483647 - 1)

extern int __finite(double);
extern int __finitef(float);
extern int __finitel(long double);
extern int __isinf(double);
extern int __isinff(float);
extern int __isnan(double);
extern int __isnanf(float);
extern int __signgam(void);
extern double copysign(double, double);
extern int finite(double);
extern double frexp(double, int *);
extern double ldexp(double, int);
extern double modf(double, double *);
extern double acos(double);
extern double asin(double);
extern double atan(double);
extern double atan2(double, double);
extern double ceil(double);
extern double cbrt(double);
extern double cos(double);
extern double cosh(double);
extern double erf(double);
extern double erfc(double);
extern double exp(double);
extern double expm1(double);
extern double fabs(double);
extern double floor(double);
extern double fmod(double, double);
extern double gamma(double);
extern double hypot(double, double);
extern int ilogb(double);
extern double j0(double);
extern double j1(double);
extern double jn(int, double);
extern double lgamma(double);
extern double log(double);
extern double log10(double);
extern double log1p(double);
extern double logb(double);
extern double nextafter(double, double);
extern double pow(double, double);
extern double remainder(double, double);
extern double rint(double);
extern double scalb(double, double);
extern double sin(double);
extern double sinh(double);
extern double sqrt(double);
extern double tan(double);
extern double tanh(double);
extern double y0(double);
extern double y1(double);
extern double yn(int, double);
extern float copysignf(float, float);
extern long double copysignl(long double, long double);
extern int finitef(float);
extern int finitel(long double);
extern float frexpf(float, int *);
extern long double frexpl(long double, int *);
extern float ldexpf(float, int);
extern long double ldexpl(long double, int);
extern float modff(float, float *);
extern long double modfl(long double, long double *);
extern float acosf(float);
extern float acoshf(float);
extern float acosl(long double);
extern float asinf(float);
extern float asinhf(float);
extern float asinhl(long double);
extern float atan2f(float, float);
extern long double atan2l(long double, long double);
extern float atanf(float);
extern float atanhf(float);
extern float atanhl(long double);
extern float cbrf(float);
extern float cbttl(long double);
extern float ceilf(float);
extern float ceill(long double);
extern float cosf(float);
extern float coshf(float);
extern float coshl(long double);
extern float dremf(float, float);
extern long double dreml(long double, long double);
extern float erfcf(float);
extern long double erfcl(long double);
extern long double erfcl(long double);
extern long double erf1(long double);
extern double exp2(double);
extern float exp2f(float);
extern long double exp2l(long double);
extern float expf(float);
extern long double expl(long double);
extern float expm1f(float);
extern long double expm1l(long double);
extern float fabsf(float);
extern long double fabsl(long double);
extern double fdim(double, double);
extern float fdimf(float, float);
extern long double fdiml(long double, long double);
extern float floorf(float);
extern long double floorl(long double);
extern double fma(double, double, double);
extern float fmaf(float, float, float);
extern long double fmal(long double, long double, long double);
extern double fmax(double, double);
extern float fmaxf(float, float);
extern long double fmaxl(long double, long double);
extern double fmin(double, double);
extern float fminf(float, float);
extern long double fminl(long double, long double);
extern float fmodf(float, float);
extern long double fmodl(long double, long double);
extern float gammaf(float);
extern long double gammal(long double);
extern float hypotf(float, float);
extern long double hypotl(long double, long double);
extern int ilogbf(float);
extern int ilogbl(long double);
extern float j0f(float);
extern long double j0l(long double);
extern float j1f(float);
extern long double j1l(long double);
extern float jnf(int, float);
extern long double jnl(int, long double);
extern double lgamma_r(double, int *);
extern float lgammaf(float);
extern float lgammaf_r(float, int *);
extern long double lgammal(long double);
extern double lgammal_r(long double, int *);
extern long long int llrint(double);
extern long long int llrintf(float);
extern long long int llrintl(long double);
extern long long int llround(double);
extern long long int llroundf(float);
extern long long int llroundl(long double);
extern float log10f(float);
extern long double log10l(long double);
extern float log1pf(float);
extern long double log1pl(long double);
extern double log2(double);
extern float log2f(float);
extern long double log2l(long double);
extern float logbf(float);
extern long double logbl(long double);
extern float logf(float);
extern long double logl(long double);
extern long int lrint(double);
extern long int lrintf(float);
extern long int lrintl(long double);
extern long int lround(double);
extern long int lroundf(float);
extern long int lroundl(long double);
extern int matherr(struct exception *);
extern double nan(const char *);
extern float nanf(const char *);
extern long double nanl(const char *);
extern double nearbyint(double);
extern float nearbyintf(float);
extern long double nearbyintl(long double);
extern double nextafter(double, long double);
extern float nextafterf(float, float);
extern long double nextafterl(long double, long double);
extern double nexttoward(double, long double);
extern float nexttowardf(float, long double);
extern long double nexttowardl(long double, long double);
extern double pow10(double);
extern float pow10f(float);
extern long double pow10l(long double);
extern float powf(float, float);
extern long double powl(long double, long double);
extern float remainderf(float, float);
extern long double remainderl(long double, long double);
extern double remquo(double, double, int *);
extern float remquof(float, float, int *);
extern long double remquol(long double, long double, int *);
extern double rint(double);
extern float roundf(float);
extern long double roundl(long double);
extern float scalbf(float, float);
extern long double scalbl(long double, long double);
extern double significand(double);
extern float significandf(float);
extern long double significandl(long double);
extern void sincos(double, double *, double *);
extern void sincosf(float, float *, float *);
extern void sincosl(long double, long double *, long double *);
extern float sinf(float);
extern float sinhf(float);
extern long double sinhl(long double);
extern long double sinh1(long double);
extern long double sin1(long double);
extern float sqrtf(float);
extern long double sqrtl(long double);
extern double tanf(float);
extern float tanhf(float);
extern long double tanhl(long double);
extern long double tanh1(long double);
extern double tgamma(double);
extern float tgammaf(float);
extern long double tanhgl(long double);
extern long double tanhgl(1, long double);
extern long double trunc(double);
extern float truncf(float);
extern long double trunc1l(long double);
extern float y0f(float);
extern long double y0l(long double);
extern float ylf(float);
extern long double yll(long double);
extern float ynf(int, float);
extern long double yn1(int, long double);
extern int __fpclassifyl(long double);
11 Libraries

extern long double exp2l(long double);
extern long double exp2l(long double);

11.6 Interface Definitions for libm

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

Other interfaces listed in Section 11.4 shall behave as described in the referenced base document.

__fpclassifyl

Name

__fpclassifyl — test for infinity

Synopsis

int __fpclassifyl(long double arg);

Description

__fpclassifyl() has the same specification as fpclassifyl() in ISO POSIX (2003), except that the argument type for __fpclassifyl() is known to be long double.

__fpclassifyl() is not in the source standard; it is only in the binary standard.

11.7 Interfaces for libpthread

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

Table 11-27 libpthread Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libpthread</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libpthread.so.0</td>
</tr>
</tbody>
</table>

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

[LFS] Large File Support
[LSB] This Specification

11.7.1 Realtime Threads

11.7.1.1 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_getinheritsched(GLIB) | pthread_attr_getchedpolicy(GLIB) | pthread_attr_getcope(GLIBC_2.0) | pthread_attr_setinheritsched(GLIBC) |
11 Libraries

<table>
<thead>
<tr>
<th>C_2.0) [SUSv3]</th>
<th>C_2.0) [SUSv3]</th>
<th>[SUSv3]</th>
<th>.2.0) [SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_setschedpolicy(GLIBC_2.0) [SUSv3]</td>
<td>pthread_attr_setschedpolicy(GLIBC_2.0) [SUSv3]</td>
<td>pthread_getschedparam(GLIBC_2.0) [SUSv3]</td>
<td>pthread_setschedparam(GLIBC_2.0) [SUSv3]</td>
</tr>
</tbody>
</table>

11.7.2 Advanced Realtime Threads

11.7.2.1 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.

11.7.3 Posix Threads

11.7.3.1 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

<table>
<thead>
<tr>
<th>_pthread_cleanup_pop(GLIBC_2.0) [LSB]</th>
<th>_pthread_cleanup_push(GLIBC_2.0) [LSB]</th>
<th>pthread_attr_destroy(GLIBC_2.0) [SUSv3]</th>
<th>pthread_attr_getdetachedstate(GLIBC_2.0) [SUSv3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_attr_getguardsize(GLIBC_2.1) [SUSv3]</td>
<td>pthread_attr_getschedparam(GLIBC_2.0) [SUSv3]</td>
<td>pthread_attr_getsetstate(GLIBC_2.2) [SUSv3]</td>
<td>pthread_attr_getsetstateaddr(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_getstacksize(GLIBC_2.1) [SUSv3]</td>
<td>pthread_attr_init(GLIBC_2.1) [SUSv3]</td>
<td>pthread_attr_setdetachedstate(GLIBC_2.0) [SUSv3]</td>
<td>pthread_attr_setgguardsize(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>pthread_attr_setschedparam(GLIBC_2.0) [SUSv3]</td>
<td>pthread_attr_setsetstate(GLIBC_2.2) [SUSv3]</td>
<td>pthread_attr_setsetstateaddr(GLIBC_2.1) [SUSv3]</td>
<td>pthread_attr_setsetstacksize(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>pthread_cancel(GLIBC_2.0) [SUSv3]</td>
<td>pthread_cond_broadcast(GLIBC_2.3.2) [SUSv3]</td>
<td>pthread_cond_destroy(GLIBC_2.3.2) [SUSv3]</td>
<td>pthread_cond_init(GLIBC_2.3.2) [SUSv3]</td>
</tr>
<tr>
<td>pthread_cond_signal(GLIBC_2.3.2) [SUSv3]</td>
<td>pthread_cond_timedwait(GLIBC_2.3.2) [SUSv3]</td>
<td>pthread_cond_wait(GLIBC_2.3.2) [SUSv3]</td>
<td>pthread_condattr_destroy(GLIBC_2.0) [SUSv3]</td>
</tr>
<tr>
<td>pthread_condattr_getpshared(GLIBC_C_2.2) [SUSv3]</td>
<td>pthread_condattr_init(GLIBC_2.0) [SUSv3]</td>
<td>pthread_condattr_setpshared(GLIBC_C_2.2) [SUSv3]</td>
<td>pthread_create(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>pthread_detach(GLIBC_2.0) [SUSv3]</td>
<td>pthread_equal(GLIBC_2.0) [SUSv3]</td>
<td>pthread_exit(GLIBC_2.0) [SUSv3]</td>
<td>pthread_getcontext(GLIBC_2.1) [SUSv3]</td>
</tr>
<tr>
<td>pthread_getspecific(GLIBC_2.0) [SUSv3]</td>
<td>pthread_join(GLIBC_2.0) [SUSv3]</td>
<td>pthread_key_create(GLIBC_2.0) [SUSv3]</td>
<td>pthread_key_delete(GLIBC_2.0) [SUSv3]</td>
</tr>
</tbody>
</table>
### 11.7.4 Thread aware versions of libc interfaces

#### 11.7.4.1 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**

<table>
<thead>
<tr>
<th>Function</th>
<th>GLIBC_2.0</th>
<th>GLIBC_2.2</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>lseek64</td>
<td>[GLIBC_2.2]</td>
<td>[LFS]</td>
<td></td>
</tr>
<tr>
<td>open64</td>
<td>[GLIBC_2.2]</td>
<td>[LFS]</td>
<td></td>
</tr>
<tr>
<td>pread</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pread64</td>
<td>[GLIBC_2.2]</td>
<td>[LFS]</td>
<td></td>
</tr>
<tr>
<td>pwrite</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pwrite64</td>
<td>[GLIBC_2.2]</td>
<td>[LFS]</td>
<td></td>
</tr>
</tbody>
</table>

---

**11 Libraries**

<table>
<thead>
<tr>
<th>Function</th>
<th>GLIBC_2.0</th>
<th>GLIBC_2.2</th>
<th>SUSv3</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_kill</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_destroy</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_init</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_lock</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_trylock</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pthread_mutex_unlock</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_destroy</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_getpshared</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_gettype</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_init</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_setpshared</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_mutexattr_settype</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_once</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_destroy</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_init</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_rdlock</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_wrlock</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_timedrdlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock timedwrlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_unlock</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_trimedrlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_trimedwrlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_tryrdlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_trywrlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_tryrdlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlock_timedwrlock</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_destroy</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_getpshared</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_gettype</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_init</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_setpshared</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_rwlockattr_settype</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_self</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setcancel_state</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getcancel</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setspecification</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setspecification</td>
<td>[SUSv3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pthread_setcancel</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setcancel</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_setconcurrency</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_getconcurrency</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>pthread_testcancel</td>
<td>[GLIBC_2.0]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_close</td>
<td>[GLIBC_2.1.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_destroy</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_getvalue</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_open</td>
<td>[GLIBC_2.1.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_post</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_timedwait</td>
<td>[GLIBC_2.2]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_trywait</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_unlink</td>
<td>[GLIBC_2.1.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
<tr>
<td>sem_wait</td>
<td>[GLIBC_2.1]</td>
<td>[SUSv3]</td>
<td></td>
</tr>
</tbody>
</table>
11.8 Data Definitions for libpthread

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

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

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

11.8.1 pthard.h

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

89
11 Libraries

```c
void *__stackaddr;
unsigned long int __stacksize;

pthread_attr_t *, struct
sched_param {
    int sched_priority;
}

*)

extern int pthread_attr_getschedpolicy(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_getscope(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_init(pthread_attr_t *);

extern int pthread_attr_setdetachstate(pthread_attr_t *, int);
extern int pthread_attr_setinheritsched(pthread_attr_t *, int);
extern int pthread_attr_setschedparam(pthread_attr_t *, const struct
    sched_param {
        int sched_priority;
    }

*)

extern int pthread_attr_setschedpolicy(pthread_attr_t *, int);
extern int pthread_attr_setscopescope(pthread_attr_t *, int);
extern int pthread_cancel(typedef unsigned long int pthread_t);
extern int pthread_cond_broadcast(pthread_cond_t *);
extern int pthread_cond_destroy(pthread_cond_t *);
extern int pthread_cond_init(pthread_cond_t *, const typedef struct {
    int __dummy;
} pthread_condattr_t *);

extern int pthread_cond_signal(pthread_cond_t *);
extern int pthread_cond_timedwait(pthread_cond_t *, pthread_mutex_t *,
    const struct timespec {
        time_t tv_sec; long int tv_nsec;
    }

*)

extern int pthread_cond_wait(pthread_cond_t *, pthread_mutex_t *);
extern int pthread_condattr_destroy(pthread_condattr_t *);
extern int pthread_condattr_init(pthread_condattr_t *);
extern int pthread_create(pthread_t *, const typedef struct {
    int __detachstate;
    int __schedpolicy;
    struct sched_param __schedparam;
    int __inheritsched;
    int __scope;
    size_t __guardsize;
    int __stackaddr_set;
```
11 Libraries

```c
void *__stackaddr;
unsigned long int __stacksize;
pthread_attr_t *;
void *__(*__start_routine) (void *p1)
  , void *);

extern int pthread_detach(typedef unsigned long int pthread_t);
extern int pthread_equal(typedef unsigned long int pthread_t,
  typedef unsigned long int pthread_t);
extern void pthread_exit(void *);
extern int pthread_getschedparam(typedef unsigned long int pthread_t,
  int *, struct sched_param {
    int sched_priority;}
  *
  );
extern void *__pthread_getspecific(typedef unsigned int pthread_key_t);
extern int pthread_join(typedef unsigned long int pthread_t, void **);
extern int pthread_key_create(pthread_key_t *, void (*destr_func) (void *
  )
  )
  );
extern int pthread_key_delete(typedef unsigned int pthread_key_t);
extern int pthread_mutex_destroy(pthread_mutex_t *);
extern int pthread_mutex_init(pthread_mutex_t *, const typedef struct {
  int __mutexkind;
  pthread_mutexattr_t *;
  }
  pthread_mutexattr_t *
  );
extern int pthread_mutex_lock(pthread_mutex_t *);
extern int pthread_mutex_trylock(pthread_mutex_t *);
extern int pthread_mutex_unlock(pthread_mutex_t *);
extern int pthread_mutexattr_destroy(pthread_mutexattr_t *
  );
extern int pthread_mutexattr_init(pthread_mutexattr_t *
  );
extern int pthread_once(pthread_once_t *, void (*init_routine) (void) *
  )
  );
extern int pthread_rwlock_destroy(pthread_rwlock_t *);
extern int pthread_rwlock_init(pthread_rwlock_t *, pthread_rwlockattr_t *
  );
extern int pthread_rwlock_rdlock(pthread_rwlock_t *);
extern int pthread_rwlock_tryrdlock(pthread_rwlock_t *
  );
extern int pthread_rwlock_trywrlock(pthread_rwlock_t *
  );
extern int pthread_rwlock_unlock(pthread_rwlock_t *);
extern int pthread_rwlock_wrlock(pthread_rwlock_t *);
extern int pthread_rwlockattr_destroy(pthread_rwlockattr_t *
  );
extern int pthread_rwlockattr_getpshared(const typedef struct {
  int __lockkind;
  int __pshared;
  pthread_rwlockattr_t *, int *
  *
  );
extern int pthread_rwlockattr_init(pthread_rwlockattr_t *
  );
extern typedef unsigned long int pthread_t pthread_self(void);
extern int pthread_setcancelstate(int, int *);
extern int pthread_setcanceltype(int, int *);
extern int pthread_setschedparam(typedef unsigned long int pthread_t,
  int, const struct sched_param {
    int sched_priority;}
  *
  );
extern int pthread_setspecific(typedef unsigned int pthread_key_t,
  const void *);
extern void pthread_testcancel(void);
extern int pthread_attr_getguardsize(const typedef struct {
  int __detachstate;
  int __schedpolicy;
  struct sched_param __schedparam;
```
typedef unsigned long int

int __inheritsched;
int __scope;
size_t __guardsize;
int __stackaddr_set;
void *__stackaddr;
unsigned long int __stacksize;

extern int pthread_attr_setguardsize(pthread_attr_t *,
typeof(unsigned long int)

int __scope);

extern int pthread_attr_setstackaddr(pthread_attr_t *, void *);
extern int pthread_attr_getstackaddr(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;});

extern int pthread_attr_setstacksize(pthread_attr_t *,
typeof(unsigned long int)

size_t);
extern int pthread_attr_getstacksize(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;});

extern int pthread_mutexattr_gettype(const typedef struct {
int __mutexkind;});

extern int pthread_mutexattr_settype(pthread_mutexattr_t *, int);
extern int pthread_mutexattr_getpshared(const typedef struct {
int __mutexkind;});

extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
extern int pthread_getconcurrency(void);
extern int pthread_setconcurrency(int);

extern int pthread_attr_getstack(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;});

extern int pthread_attr_setstack(pthread_attr_t *, void *,
typeof(unsigned long int)

size_t *, size_t *);

extern int pthread_mutexattr_getpshared(const typedef struct {
int __mutexkind;});

extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
extern int pthread_mutexattr_getpshared(const typedef struct {
int __mutexkind;});

extern int pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
extern int pthread_rwlock_timedrdlock(pthread_rwlock_t *, const struct timespec {
    time_t tv_sec; long int tv_nsec;
} *);

extern int pthread_rwlock_timedwrlock(pthread_rwlock_t *, const struct timespec {
    time_t tv_sec; long int tv_nsec;
} *);

extern int __register_atfork(void (*prepare) (void), void (*parent) (void), void (*child) (void), void *);

extern int pthread_setschedprio(typedef unsigned long int pthread_t, int);

### 11.8.2 semaphore.h

extern int sem_close(sem_t *);
extern int sem_destroy(sem_t *);
extern int sem_getvalue(sem_t *, int *);
extern int sem_init(sem_t *, int, unsigned int);
extern sem_t *sem_open(const char *, int, ...);
extern int sem_post(sem_t *);
extern int sem_trywait(sem_t *);
extern int sem_unlink(const char *);
extern int sem_wait(sem_t *);
extern int sem_timedwait(sem_t *, const struct timespec *);

## 11.9 Interfaces for libgcc_s

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

<table>
<thead>
<tr>
<th>Library:</th>
<th>libgcc_s</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libgcc_s.so.1</td>
</tr>
</tbody>
</table>

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

[LSB] This Specification

### 11.9.1 Unwind Library

#### 11.9.1.1 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>
<thead>
<tr>
<th>_Unwind_Backtrace(GCC_3.3) [LSB]</th>
<th>_Unwind_DeleteException(GCC_3.0) [LSB]</th>
<th>_Unwind_FindEndingFunctionAddress(GCC_3.3) [LSB]</th>
<th>_Unwind_FindFrame(GCC_3.0) [LSB]</th>
</tr>
</thead>
</table>
11.10 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 ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

11.10.1 unwind.h

```c
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
    _Unwind_Stop_Fn, void *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern void _Unwind_DeclareException(struct _Unwind_Exception *);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
    _Unwind_Stop_Fn, void *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
```
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *, _Unwind_Stop_Fn, void *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *, _Unwind_Stop_Fn, void *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern fde * _Unwind_Find_FDE(void *, struct dwarf_eh_base *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *, _Unwind_Stop_Fn, void *);
extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
extern _Unwind_Word _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_RaiseException(struct _Unwind_Exception *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *, _Unwind_Stop_Fn, void *);
extern _Unwind_Word _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind.RaiseException(struct _Unwind_Exception *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *, _Unwind_Stop_Fn, void *);
extern _Unwind_Word _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind.RaiseException(struct _Unwind_Exception *);
extern void _Unwind_Resume(struct _Unwind_Exception *);
extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
extern void _Unwind_DeleteException(struct _Unwind_Exception *);
extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *, _Unwind_Stop_Fn, void *);
extern _Unwind_Word _Unwind_GetIP(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind.Resume_or_Rethrow(struct _Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind.Resume_or_Rethrow(struct _Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind.Resume_or_Rethrow(struct _Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind.Resume_or_Rethrow(struct _Unwind_Exception *);
11 Libraries

11.11 Interface Definitions for libgcc_s

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

Other interfaces listed in Section 11.9 shall behave as described in the referenced base document.

_Unwind_DeleteException

Name

_Unwind_DeleteException — private C++ error handling method

Synopsis

void _Unwind_DeleteException(struct _Unwind_Exception * object);

Description

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

_Unwind_Find_FDE

Name

_Unwind_Find_FDE — private C++ error handling method

Synopsis

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

Description

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

**Name**

_Unwind_ForcedUnwind — private C++ error handling method

**Synopsis**

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

**Description**

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

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

**Return Value**

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

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

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

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

- **_URC_FATAL_PHASE2_ERROR**
  
  The `stop()` function may return this code for other fatal conditions like stack corruption.
_Unwind_GetDataRelBase

Name

_Unwind_GetDataRelBase — private IA64 C++ error handling method

Synopsis

_Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context * context);

Description

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

_Unwind_GetGR

Name

_Unwind_GetGR — private C++ error handling method

Synopsis

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

Description

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

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

_Unwind_GetIP

Name

_Unwind_GetIP — private C++ error handling method

Synopsis

_Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context * context);

Description

_Unwind_GetIP() returns the instruction pointer value for the routine identified by the unwind context.
11 Libraries

_Unwind_GetLanguageSpecificData

Name
_Unwind_GetLanguageSpecificData — private C++ error handling method

Synopsis
_Unwind_Ptr _Unwind_GetLanguageSpecificData(struct _Unwind_Context * context, uint value);

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

_Unwind_GetRegionStart

Name
_Unwind_GetRegionStart — private C++ error handling method

Synopsis
_Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context * context);

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

_Unwind_GetTextRelBase

Name
_Unwind_GetTextRelBase — private IA64 C++ error handling method

Synopsis
_Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context * context);

Description
_Unwind_GetTextRelBase() calls the abort method, then returns.
_Unwind.RaiseException

Name

_Unwind.RaiseException — private C++ error handling method

Synopsis

_Unwind_Reason_Code _Unwind.RaiseException(struct _Unwind_Exception * object);

Description

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

Return Value

_Unwind.RaiseException() does not return unless an error condition is found. If an error condition occurs, an _Unwind_Reason_Code is returned:

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

- _URC_FATAL_PHASE1_ERROR
  The unwinder encountered an unexpected error during phase one, because of something like stack corruption. The unwind runtime will not have modified the stack. The C++ runtime will normally call terminate() in this case.

- _URC_FATAL_PHASE2_ERROR
  The unwinder encountered an unexpected error during phase two. This is usually a throw, which will call terminate().

_Unwind.Resume

Name

_Unwind.Resume — private C++ error handling method

Synopsis

void _Unwind.Resume(struct _Unwind_Exception * object);

Description

_Unwind.Resume() resumes propagation of an existing exception object. A call to this routine is inserted as the end of a landing pad that performs cleanup, but does not resume normal execution. It causes unwinding to proceed further.
11 Libraries

_Unwind_SetGR

Name

_Unwind_SetGR — private C++ error handling method

Synopsis

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

Description

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

_Unwind_SetIP

Name

_Unwind_SetIP — private C++ error handling method

Synopsis

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

Description

_Unwind_SetIP() sets the value of the instruction pointer for the routine identified
by the unwind context

11.12 Interfaces for libdl

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

Table 11-33 libdl Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libdl</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libdl.so.2</td>
</tr>
</tbody>
</table>

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

[LSB] This Specification

11.12.1 Dynamic Loader

11.12.1.1 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(GLIBC_2.0) [LSB] | dlclose(GLIBC_2.0) [SUSv3] | dlerror(GLIBC_2.0) [SUSv3] | dlopen(GLIBC_2.1) [LSB] |
11 Libraries

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

11.13.1 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 *);

11.14 Interfaces for libcrypt

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

Table 11-35 libcrypt Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libcrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libcrypt.so.1</td>
</tr>
</tbody>
</table>

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


11.14.1 Encryption

11.14.1.1 Interfaces for Encryption

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

Table 11-36 libcrypt - Encryption Function Interfaces

<table>
<thead>
<tr>
<th>crypt(GLIBC_2.0)</th>
<th>encrypt(GLIBC_2.0) [SUSv3]</th>
<th>setkey(GLIBC_2.0) [SUSv3]</th>
</tr>
</thead>
</table>

103
IV Utility Libraries
12 Libraries

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

12.1 Interfaces for libz

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

<table>
<thead>
<tr>
<th>Library</th>
<th>libz</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libz.so.1</td>
</tr>
</tbody>
</table>

12.1.1 Compression Library

12.1.1.1 Interfaces for Compression Library

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

12.2 Data Definitions for libz

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

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

This specification uses the ISO C (1999) C Language as the reference programming
language, and data definitions are specified in ISO C. 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.

12.2.1 zlib.h

extern int gzread(gzFile, voidp, unsigned int);
extern int gzclose(gzFile);
extern gzFile gzopen(const char *, const char *);
extern gzFile gzdopen(int, const char *);
extern int gzwrite(gzFile, voidpc, unsigned int);
extern int gzflush(gzFile, int);
extern const char *gzerror(gzFile, int *);
extern uLong adler32(uLong, const Bytef *, uInt);
extern int compress(Bytef *, uLongf *, const Bytef *, uLong);
extern int compress2(Bytef *, uLongf *, const Bytef *, uLong, int);
extern uLong crc32(uLong, const Bytef *, uInt);
extern int deflate(z_streamp, int);
extern int deflateCopy(z_streamp, z_streamp);
extern int deflateEnd(z_streamp);
extern int deflateInit2_(z_streamp, int, int, int, int, const char *
  int);
extern int deflateInit_(z_streamp, int, const char *, int);
extern int deflateParams(z_streamp, int, int);
extern int deflateReset(z_streamp);
extern int deflateSetDictionary(z_streamp, const Bytef *, uInt);
extern const uLongf *get_crc_table(void);
extern int gzeof(gzFile);
extern int gzgetc(gzFile);
extern char *gzgets(gzFile, char *, int);
extern int gzputc(gzFile, int);
extern int gzputs(gzFile, const char *);
extern int gzrewind(gzFile);
extern int gzsetparams(gzFile, int, int);
extern z_off_t gzseek(gzFile, z_off_t, int);
extern int gzprintf(gzFile, const char *, ...);
extern int inflate(z_streamp, int);
extern int inflateEnd(z_streamp);
extern int inflateInit2_(z_streamp, int, const char *, int);
extern int inflateInit_(z_streamp, const char *, int);
extern int inflateReset(z_streamp);
extern int inflateSetDictionary(z_streamp, const Bytef *, uInt);
extern int inflateSync(z_streamp);
extern int inflateSyncPoint(z_streamp);
extern int uncompress(Bytef *, uLongf *, const Bytef *, uLong);
extern const char *zError(int);
extern const char *zlibVersion(void);
extern uLong deflateBound(z_streamp, uLong);
extern uLong compressBound(uLong);

12 Libraries

12.3 Interfaces for libncurses

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

Table 12-2 libncurses Definition

<table>
<thead>
<tr>
<th>Library:</th>
<th>libncurses</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libncurses.so.5</td>
</tr>
</tbody>
</table>

12.3.1 Curses

12.3.1.1 Interfaces for Curses

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

12.4 Data Definitions for libncurses

This section defines global identifiers and their values that are associated with interfaces contained in libncurses. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.
This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C. 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.

12.4.1 curses.h

extern int addch(const chtype);
extern int addchnstr(const chtype *, int);
extern int addchstr(const chtype *);
extern int addnstr(const char *, int);
extern int addstr(const char *);
extern int attroff(int);
extern int attron(int);
extern int attrset(int);
extern int attr_get(attr_t *, short *, void *);
extern int attr_off(attr_t, void *);
extern int attr_on(attr_t, void *);
extern int attr_set(attr_t, short, void *);
extern int baudrate(void);
extern int beep(void);
extern int bkgd(chtype);
extern void bkgdset(chtype);
extern int border(chtype, chtype, chtype, chtype, chtype, chtype,
    chtype, chtype);
extern int box(WINDOW *, chtype, chtype);
extern bool can_change_color(void);
extern int cbreak(void);
extern int chgat(int, attr_t, short, const void *);
extern int clear(void);
extern int clearok(WINDOW *, bool);
extern int clrtobot(void);
extern int clrtoeol(void);
extern int color_content(short, short *, short *, short *);
extern int color_set(short, void *);
extern int copywin(const WINDOW *, WINDOW *, int, int, int, int, int,
    int);
extern int curs_set(int);
extern int def_prog_mode(void);
extern int def_shell_mode(void);
extern int delay_output(int);
extern int delch(void);
extern void delscreen(SCREEN *);
extern int delwin(WINDOW *);
extern int deleteIn(void);
extern int WINDOW *(derwin(WINDOW *, int, int, int, int, int, int, int));
extern int dopupdate(void);
extern int WINDOW *(dupwin(WINDOW *));
extern int echo(void);
extern int echochar(const chtype);
extern int erase(void);
extern int endwin(void);
extern char erasechar(void);
extern void filter(void);
extern int flash(void);
extern int flushinp(void);
extern chtype getbkgd(WINDOW *);
extern int getch(void);
extern int getnstr(char *, int);
extern int getstr(char *);
extern WINDOW *getwin(FILE *);
extern int halfdelay(int);
extern bool has_colors(void);
extern bool has_ic(void);
extern bool has_il(void);
extern int hline(chtype, int);
extern void idcok(WINDOW *, bool);
extern int idlok(WINDOW *, bool);
extern void immedok(WINDOW *, bool);
extern chtype inch(void);
extern int inchnstr(chtype *, int);
extern int inchstr(chtype *);
extern WINDOW *initscr(void);
extern int init_color(short, short, short, short);
extern int init_pair(short, short, short);
extern int innstr(char *, int);
extern int insch(chtype);
extern int insdelln(int);
extern int insertln(void);
extern int insnstr(const char *, int);
extern int insstr(const char *);
extern int instr(char *);
extern int intrflush(WINDOW *, bool);
extern bool isendwin(void);
extern bool is_linetouched(WINDOW *, int);
extern bool is_wintouched(WINDOW *);
extern const char *keyname(int);
extern int keypad(WINDOW *, bool);
extern char killchar(void);
extern int leaveok(WINDOW *, bool);
extern char *longname(void);
extern int meta(WINDOW *, bool);
extern int move(int, int);
extern int mvaddch(int, int, const chtype);
extern int mvaddchnstr(int, int, const chtype *, int);
extern int mvaddchstr(int, int, const chtype *);
extern int mvaddnstr(int, int, const char *, int);
extern int mvaddstr(int, int, const char *);
extern int mvchgat(int, int, int, attr_t, short, const void *);
extern int mvcur(int, int, int, int);
extern int mvdelch(int, int);
extern int mvderwin(WINDOW *, int, int);
extern int mvgetnstr(int, int, char *, int);
extern int mvgetstr(int, int, char *, int);
extern int mvhline(int, int, chtype, int);
extern chtype mvinch(int, int);
extern int mvinchnstr(int, int, chtype *, int);
extern int mvinchstr(int, int, chtype *);
extern int mvinnstr(int, int, char *, int);
extern int mvinsch(int, int, chtype);
extern int mvinsnstr(int, int, const char *, int);
extern int mvinnstr(int, int, const char *);
extern int mvprintw(int, int, char *, ...);
extern int mvscanw(int, int, const char *, ...);
extern int mvvline(int, int, chtype, int);
extern int mvwaddch(WINDOW *, int, int, const chtype);
extern int mvwaddchnstr(WINDOW *, int, int, const chtype *, int);
extern int mvwaddchstr(WINDOW *, int, int, const chtype *);
extern int mvwaddnstr(WINDOW *, int, int, const char *, int);
extern int mvwaddstr(WINDOW *, int, int, const char *);
extern int mvwchgat(WINDOW *, int, int, int, attr_t, short, const void *);
extern int mvwdelch(WINDOW *, int, int);
extern int mvwgetch(WINDOW *, int, int);
extern int mvwgetnstr(WINDOW *, int, int, char *, int);
extern int mvwgetstr(WINDOW *, int, int, char *);
extern int mvwhline(WINDOW *, int, int, chtype, int);
extern int mvwin(WINDOW *, int, int);
extern chtype mvwinch(WINDOW *, int, int);
extern int mvwinchnstr(WINDOW *, int, int, chtype *, int);
extern int mvwinchstr(WINDOW *, int, int, chtype *);
extern int mvwinsch(WINDOW *, int, int, chtype);
extern int mvwinsnstr(WINDOW *, int, int, const char *, int);
extern int mvwinsstr(WINDOW *, int, int, const char *);
extern int mvwprintw(WINDOW *, int, int, char *, ...);
extern int mvwscnwcstr(WINDOW *, int, int, char *, int);
extern int mvwvline(WINDOW *, int, int, chtype, int);
extern int napms(int);
extern WINDOW *newpad(int, int);
extern SCREEN *newterm(const char *, FILE *, FILE *);
extern WINDOW *newwin(int, int, int, int);
extern int nl(void);
extern int nocbreak(void);
extern int nodelay(WINDOW *, bool);
extern int noecho(void);
extern int nonl(void);
extern void noqiflush(void);
extern int noraw(void);
extern int notimeout(WINDOW *, bool);
extern int overlay(const WINDOW *, WINDOW *);
extern int overwrite(const WINDOW *, WINDOW *);
extern int pair_content(short, short *, short *);
extern int pechochar(WINDOW *, chtype);
extern int pnoutrefresh(WINDOW *, int, int, int, int, int, int);
extern int prefresh(WINDOW *, int, int, int, int, int, int);
extern int printw(char *, ...);
extern int putwin(WINDOW *, FILE *);
extern void qiflush(void);
extern int raw(void);
extern int redrawwin(WINDOW *);
extern int refresh(void);
extern int resetty(void);
extern int reset_prog_mode(void);
extern int reset_shell_mode(void);
extern int ripoffline(int, int (*init) (WINDOW *, int));
extern int savetty(void);
extern int scanw(const char *, ...);
extern int scr_dump(const char *);
extern int scr_init(const char *);
extern int scrl(int);
extern int scroll(WINDOW *);
extern int scrollok(WINDOW *, bool);
extern int scr_restore(const char *);
extern int scr_set(const char *);
extern int scr_setreg(int, int);
extern int setscrreg(int, int);
extern int set_term(SCREEN *);
extern int slk_attroff(const chtype);
extern int slk_attron(const chtype);
extern int slk_attrset(const typedef unsigned long int chtype);
extern int slk_attr_set(const typedef chtype attr_t, short, void *);
extern int slk_clear(void);
extern int slk_color(short);
extern int slk_init(int);
extern char *slk_label(int);
extern int slk_noutrefresh(void);
extern int slk_refresh(void);
extern int slk_restore(void);
extern int slk_set(int, const char *, int);
extern int slk_touch(void);
extern int standout(void);
extern int standend(void);
extern int start_color(void);
extern WINDOW *subpad(WINDOW *, int, int, int, int);
extern WINDOW *subwin(WINDOW *, int, int, int, int);
extern int syncok(WINDOW *, typedef unsigned char bool);
extern int slk_color(short);
extern int slk_init(int);
extern char *slk_label(int);
extern int slk_noutrefresh(void);
extern int slk_refresh(void);
extern int slk_restore(void);
extern int slk_touch(void);
extern int standout(void);
extern int standend(void);
extern int start_color(void);
extern char *termname(void);
extern void timeout(int);
extern int typeahead(int);
extern int ungetch(int);
extern int waddchnstr(WINDOW *, const typedef unsigned long int chtype *);
extern int waddchstr(WINDOW *, const typedef unsigned long int chtype *);
extern int waddstr(WINDOW *, const char *);
extern int wattron(WINDOW *, int);
extern int wattroff(WINDOW *, int);
extern int wattrset(WINDOW *, int);
extern int wattr_get(WINDOW *, attr_t *, short *, void *);
extern void wbkgdset(WINDOW *, typedef unsigned long int chtype);
extern int wbkgd(WINDOW *, typedef unsigned long int chtype);
extern int wbkgdset(WINDOW *, typedef unsigned long int chtype);
extern int wborder(WINDOW *, typedef unsigned long int chtype,
    typedef unsigned long int chtype,
    typedef unsigned long int chtype,
    typedef unsigned long int chtype,
    typedef unsigned long int chtype,
    typedef unsigned long int chtype,
    typedef unsigned long int chtype,
    typedef unsigned long int chtype);
extern void wcursyncup(WINDOW *);
extern int wdelch(WINDOW *);
extern int wdeleteln(WINDOW *);
extern int wechochar(WINDOW *, const typedef unsigned long int chtype);
extern int werase(WINDOW *);
extern int wgetch(WINDOW *);
extern int wgetnstr(WINDOW *, char *, int);
extern int wgetstr(WINDOW *, char *);
extern int whline(WINDOW *, typedef unsigned long int chtype, int);
extern typedef unsigned long int chtype winch(WINDOW *);
extern int winchnstr(WINDOW *, chtype *, int);
extern int winchstr(WINDOW *, chtype *);
extern int winsch(WINDOW *, typedef unsigned long int chtype);
extern int winsdelln(WINDOW *, int);
extern int winsertln(WINDOW *);
extern int winsnstr(WINDOW *, const char *, int);
extern int winsstr(WINDOW *, const char *);
extern int winstr(WINDOW *, char *);
extern int wmove(WINDOW *, int, int);
extern int wnoutrefresh(WINDOW *);
extern int wprintw(WINDOW *, char *, ...);
extern int wrefresh(WINDOW *);
extern int wscanw(WINDOW *, const char *, ...);
extern int wscrl(WINDOW *, int);
extern int wsetscrreg(WINDOW *, int, int);
extern int wstandout(WINDOW *);
extern int wstandend(WINDOW *);
extern void wsyncdown(WINDOW *);
extern void wsyncup(WINDOW *);
extern void wtimeout(WINDOW *, int);
extern int wtouchln(WINDOW *, int, int, int);
extern int wvline(WINDOW *, typedef unsigned long int chtype, int);
extern char *unctrl(typedef unsigned long int chtype);
extern int COLORS(void);
extern int COLOR_PAIRS(void);
extern chtype acs_map(void);
extern WINDOW *curscr(void);
extern WINDOW *stdscr(void);
extern int COLS(void);
extern int LINES(void);
extern int touchline(WINDOW *, int, int);
extern int touchwin(WINDOW *);

12.4.2 term.h

extern int putp(const char *);
extern int tigetflag(const char *);
extern int tigetnum(const char *);
extern char *tigetstr(const char *);
extern char *tparm(const char *, ...);
extern TERMINAL *set_curterm(TERMINAL *);
extern int del_curterm(TERMINAL *);
extern int restartterm(char *, int, int *);
extern int setupterm(char *, int, int *);
extern int tgetflag(char *);
extern int tgetnum(char *);
extern int tgetflag(char *);
extern int tgetstr(char *, char **);
extern char *tgoto(const char *, int, int);
extern int tgetent(char *, const char *);
extern int tgetflag(char *);
extern int tgetnum(char *);
extern int tputs(const char *, int, int (*putcproc) (int) );
12 Libraries

```c
extern TERMINAL *cur_term(void);
```

## 12.5 Interfaces for libutil

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

<table>
<thead>
<tr>
<th>Library:</th>
<th>libutil</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONAME:</td>
<td>libutil.so.1</td>
</tr>
</tbody>
</table>

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

- [LSB] This Specification

### 12.5.1 Utility Functions

#### 12.5.1.1 Interfaces for Utility Functions

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

<table>
<thead>
<tr>
<th>Function Interfaces</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>forkpty(GLIBC_2.0) [LSB]</td>
<td>login(GLIBC_2.0) [LSB]</td>
</tr>
<tr>
<td>logwtmp(GLIBC_2.0) [LSB]</td>
<td>login_tty(GLIBC_2.0) [LSB]</td>
</tr>
<tr>
<td>logwtmp(GLIBC_2.0) [LSB]</td>
<td>logout(GLIBC_2.0) [LSB]</td>
</tr>
</tbody>
</table>
V Package Format and Installation
13 Software Installation

13.1 Package Dependencies

The LSB runtime environment shall provide the following dependencies.

- lsb-core-ia32
  
  This dependency is used to indicate that the application is dependent on features contained in the LSB-Core specification.
  
  These dependencies shall have a version of 3.0.

- Other LSB modules may add additional dependencies; such dependencies shall have the format lsb-module-ia32.

13.2 Package Architecture Considerations

- All packages must specify an architecture of i486. A LSB runtime environment must accept an architecture of i486 even if the native architecture is different.

- The archnum value in the Lead Section shall be 0x0001.
Annex A Alphabetical Listing of Interfaces

A.1 libgcc_s

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

This Specification [LSB]

Table A-1 libgcc_s Function Interfaces

<table>
<thead>
<tr>
<th>Function Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Unwind_Backtrace[LSB]</td>
</tr>
<tr>
<td>_Unwind_DeleteException[LSB]</td>
</tr>
<tr>
<td>_Unwind_FindEnclosingFunction[LSB]</td>
</tr>
<tr>
<td>_Unwind_Find_FDE[LSB]</td>
</tr>
<tr>
<td>_Unwind_ForecUnwind[LSB]</td>
</tr>
<tr>
<td>_Unwind_GetCFA[LSB]</td>
</tr>
<tr>
<td>_Unwind_GetDataRelBase[LSB]</td>
</tr>
<tr>
<td>_Unwind_GetGR[LSB]</td>
</tr>
<tr>
<td>_Unwind_GetIP[LSB]</td>
</tr>
<tr>
<td>_Unwind_GetLanguageSpecificData[LSB]</td>
</tr>
<tr>
<td>_Unwind_RaiseException[LSB]</td>
</tr>
<tr>
<td>_Unwind_Resum[LSB]</td>
</tr>
<tr>
<td>_Unwind_RaiseExcept[LSB]</td>
</tr>
<tr>
<td>_Unwind_Rerethrow[LSB]</td>
</tr>
<tr>
<td>_Unwind_SetGR[LSB]</td>
</tr>
<tr>
<td>_Unwind_SetIP[LSB]</td>
</tr>
<tr>
<td>_Unwind_GetTextRelBase[LSB]</td>
</tr>
</tbody>
</table>

A.2 libm

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

ISO C (1999) [ISOC99]

This Specification [LSB]

ISO POSIX (2003) [SUSv3]

Table A-2 libm Function Interfaces

<table>
<thead>
<tr>
<th>Function Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>_fpclassifyl[LSB]</td>
</tr>
<tr>
<td>_signbitl[ISOC99]</td>
</tr>
<tr>
<td>exp2l[SUSv3]</td>
</tr>
</tbody>
</table>
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. Secondarily, 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.
A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

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

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

**B.3 VERBATIM COPYING**

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

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

**B.4 COPYING IN QUANTITY**

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

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

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

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

**B.5 MODIFICATIONS**

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

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

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

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

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

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

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

B.6 COMBINING DOCUMENTS

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

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

B.7 COLLECTIONS OF DOCUMENTS

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

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

B.8 AGGREGATION WITH INDEPENDENT WORKS

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

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

B.9 TRANSLATION

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

B.10 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
B.11 FUTURE REVISIONS OF THIS LICENSE

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

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

B.12 How to use this License for your documents

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

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

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

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