Linux Standard Base Core Specification for S390X 3.1

Linux Standard Base Core Specification for S390X 3.1

Copyright © 2004, 2005 Free Standards Group

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

Portions of the text are copyrighted by the following parties:

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

These excerpts are being used in accordance with their respective licenses.
Linux is a trademark of Linus Torvalds.
UNIX a registered trademark of the Open Group in the United States and other countries.
LSB is a trademark of the Free Standards Group in the USA and other countries.
AMD is a trademark of Advanced Micro Devices, Inc.
Intel and Itanium are registered trademarks and Intel386 is a trademarks of Intel Corporation.
PowerPC and PowerPC Architecture are trademarks of the IBM Corporation.
OpenGL is a registered trademark of Silicon Graphics, Inc.

Contents

List of Tables

Foreword

This is version 3.1 of the Linux Standard Base Core Specification for S390X. 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 S390X 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

Name	Title	URL
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEC 60559/IEEE 754 Floating Point	IEC 60559:1989 Binary floating-point arithmetic for microprocessor systems	http://www.ieee.org/
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
ISO POSIX (2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale	http://www.unix.org/version3/

Name	Title	URL
	Including Technical Cor. 1: 2004	
ISO/IEC 14882: 2003 C++ Language	ISO/IEC 14882: 2003 Programming languages --C++	
Itanium C++ ABI	Itanium C++ ABI (Revision 1.83)	http://refspecs.freestandards.org/cxxabi-1.83.html
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
LINUX for zSeries Application Binary Interface Supplement	LINUX for zSeries Application Binary Interface Supplement	http://oss.software.ibm.com/linux390/documentation-2.2.shtml
SUSv2	CAE Specification, January 1997, System Interfaces and Headers (XSH),Issue 5 (ISBN: 1-85912-181-0, C606)	http://www.opengroup.org/publications/catalog/un.htm
SUSv2 Commands and Utilities	The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/publications/catalog/un.htm
SVID Issue 3	American Telephone and Telegraph Company, System V Interface Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989.(ISBN 0201566524)	
SVID Issue 4	System V Interface Definition,Fourth Edition	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610),	http://www.opengroup.org/publications/catalog/un.htm

Name	Title	URL
	plus Corrigendum U018	
z/Architecture Principles of Operation	z/Architecture Principles of Operation	http://oss.software.ibm.com/linux390/documentation-2.2.shtml

2.2 Informative References/Bibliography

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

Table 2-2 Other References

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

Name	Title	URL
Message-Digest Algorithm	Message-Digest Algorithm	/rfc1321.txt
RFC 1831/1832 RPC & XDR	IETF RFC 1831 & 1832	http://www.ietf.org/
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1950: ZLIB Compressed Data Format Specication	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821:Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822:Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 791:Internet Protocol	IETF RFC 791: Internet Protocol Specification	http://www.ietf.org/rfc/rfc791.txt
RPM Package Format	RPM Package Format V3.0	http://www.rpm.org/max-rpm/s1-rpm-file-format-rpm-file-format.html
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

16

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on S390X 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

Library	Runtime Name
libm	libm.so.6
libdl	libdl.so.2
libcrypt	libcrypt.so.1
libz	libz.so.1
libncurses	libncurses.so.5
libutil	libutil.so.1
libc	libc.so.6
libpthread	libpthread.so.0
proginterp	/lib64/ld-lsb-s390x.so.3
libgcc_s	libgcc_s.so.1

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

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.

- 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 possibilty 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:

name
: 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 LINUX for zSeries Application Binary Interface 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 z/Architecture is specified by the following documents

- LINUX for zSeries Application Binary Interface Supplement
- z/Architecture Principles of Operation

Only the non optional features of z/Architecture 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 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 must provide feedback to the user if a feature that is required for correct execution of the application is not present. Applications conforming to this specification should attempt to execute in a diminished capacity if a required instruction set feature is not present.

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

8.1.2 Data Representation

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

8.1.2.1 Byte Ordering

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.1.2.2 Fundamental Types

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.1.2.3 Aggregates and Unions

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.1.2.4 Bit Fields

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.2 Function Calling Sequence

LSB-conforming applications shall use the function calling sequence as defined in Chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.2.1 Registers

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.2.2 Stack Frame

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.2.3 Parameter Passing

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.2.4 Variable Argument Lists

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.2.5 Return Values

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3 Operating System Interface

LSB-conforming applications shall use the Operating System Interfaces as defined in Chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.1 Virtual Address Space

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.2 Page Size

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.3 Virtual Address Assignments

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.4 Managing the Process Stack

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.5 Coding Guidelines

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.6 Processor Execution Mode

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.7 Exception Interface

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.8 Signal Delivery

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.3.8.1 Signal Handler Interface

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.4 Process Initialization

LSB-conforming applications shall use the Process Initialization as defined in Chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.4.1 Registers

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.4.2 Process Stack

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5 Coding Examples

LSB-conforming applications may implement fundamental operations using the Coding Examples as defined in Chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5.1 Code Model Overview

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5.2 Function Prolog and Epilog

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5.3 Profiling

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5.4 Data Objects

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5.5 Function Calls

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.5.6 Dynamic Stack Space Allocation

See chapter 1 of the LINUX for zSeries Application Binary Interface Supplement.

8.6 Debug Information

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

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 , LINUX for zSeries Application Binary Interface Supplement and as supplemented by the generic LSB and this document.

9.2 ELF Header

9.2.1 Machine Information

LSB-conforming applications shall use the Machine Information as defined in Chapter 2 of the LINUX for zSeries Application Binary Interface Supplement.

9.3 Sections

See chapter 2 of the LINUX for zSeries Application Binary Interface Supplement.

9.3.1 Special Sections

The following sections are defined in the LINUX for zSeries Application Binary Interface Supplement.

Table 9-1 ELF Special Sections

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

.got

This section holds the global offset table

.plt

This section holds the procedure linkage table

9.3.2 Linux Special Sections

The following Linux S/390 specific sections are defined here.

Table 9-2 Additional Special Sections

Name	Type	Attributes
.rela.dyn	SHT_RELA	SHF_ALLOC
.rela.plt	SHT_RELA	SHF_ALLOC
.sbss	SHT_PROGBITS	SHF_WRITE

.rela.dyn

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

.rela.plt

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

.sbss

This section holds uninitialized data that contribute to the program's memory image. The system initializes the data with zeroes when the program begins to run.

9.4 Symbol Table

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

9.5 Relocation

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

9.5.1 Relocation Types

See chapter 2 of the LINUX for zSeries Application Binary Interface Supplement.

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 , LINUX for zSeries Application Binary Interface Supplement and as supplemented by the This Specification and this document.

10.2 Program Loading

See Chapter 3 of the LINUX for zSeries Application Binary Interface Supplement.

10.3 Dynamic Linking

See Chapter 3 of the LINUX for zSeries Application Binary Interface Supplement.

10.3.1 Dynamic Section

The following dynamic entries are defined in the LINUX for zSeries Application Binary Interface Supplement.

DT_JMPREL

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

DT_PLTGOT

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

10.3.2 Global Offset Table

See Chapter 3 of the LINUX for zSeries Application Binary Interface Supplement.

10.3.3 Function Addresses

See chapter 3 of the LINUX for zSeries Application Binary Interface Supplement.

10.3.4 Procedure Linkage Table

See chapter 3 of the LINUX for zSeries Application Binary Interface Supplement.

III Base Libraries

11 Libraries

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

Only those interfaces that are unique to the z/Architecture 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 `/lib64/ld-lsb-s390x.so.3`.

11.2 Interfaces for libc

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

Table 11-1 libc Definition

Library:	libc
SONAME:	libc.so.6

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

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

11.2.1 RPC

11.2.1.1 Interfaces for RPC

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 11-2 libc - RPC Function Interfaces

authnone_create(GLIBC_2.2) [SVID.4]	clnt_create(GLIBC_2.2) [SVID.4]	clnt_pcreateerror(GLIBC_2.2) [SVID.4]	clnt_perrno(GLIBC_2.2) [SVID.4]
clnt_perror(GLIBC_2.2) [SVID.4]	clnt_spcreateerror(GLIBC_2.2) [SVID.4]	clnt_sperrno(GLIBC_2.2) [SVID.4]	clnt_sperror(GLIBC_2.2) [SVID.4]
key_decryptsession(GLIBC_2.2) [SVID.3]	pmap_getport(GLIBC_2.2) [LSB]	pmap_set(GLIBC_2.2) [LSB]	pmap_unset(GLIBC_2.2) [LSB]
svc_getreqset(GLI	svc_register(GLIB	svc_run(GLIBC_2.	svc_sendreply(GL

BC_2.2) [SVID.3]	C_2.2) [LSB]	2) [LSB]	IBC_2.2) [LSB]
svcerr_auth(GLIBC_2.2) [SVID.3]	svcerr_decode(GLIBC_2.2) [SVID.3]	svcerr_noproc(GLIBC_2.2) [SVID.3]	svcerr_noprog(GLIBC_2.2) [SVID.3]
svcerr_progvers(GLIBC_2.2) [SVID.3]	svcerr_systemerr(GLIBC_2.2) [SVID.3]	svcerr_weakauth(GLIBC_2.2) [SVID.3]	svctcp_create(GLIBC_2.2) [LSB]
svcudp_create(GLIBC_2.2) [LSB]	xdr_accepted_reply(GLIBC_2.2) [SVID.3]	xdr_array(GLIBC_2.2) [SVID.3]	xdr_bool(GLIBC_2.2) [SVID.3]
xdr_bytes(GLIBC_2.2) [SVID.3]	xdr_callhdr(GLIBC_2.2) [SVID.3]	xdr_callmsg(GLIBC_2.2) [SVID.3]	xdr_char(GLIBC_2.2) [SVID.3]
xdr_double(GLIBC_2.2) [SVID.3]	xdr_enum(GLIBC_2.2) [SVID.3]	xdr_float(GLIBC_2.2) [SVID.3]	xdr_free(GLIBC_2.2) [SVID.3]
xdr_int(GLIBC_2.2) [SVID.3]	xdr_long(GLIBC_2.2) [SVID.3]	xdr_opaque(GLIBC_2.2) [SVID.3]	xdr_opaque_auth(GLIBC_2.2) [SVID.3]
xdr_pointer(GLIBC_2.2) [SVID.3]	xdr_reference(GLIBC_2.2) [SVID.3]	xdr_rejected_reply(GLIBC_2.2) [SVID.3]	xdr_replymsg(GLIBC_2.2) [SVID.3]
xdr_short(GLIBC_2.2) [SVID.3]	xdr_string(GLIBC_2.2) [SVID.3]	xdr_u_char(GLIBC_2.2) [SVID.3]	xdr_u_int(GLIBC_2.2) [LSB]
xdr_u_long(GLIBC_2.2) [SVID.3]	xdr_u_short(GLIBC_2.2) [SVID.3]	xdr_union(GLIBC_2.2) [SVID.3]	xdr_vector(GLIBC_2.2) [SVID.3]
xdr_void(GLIBC_2.2) [SVID.3]	xdr_wrapstring(GLIBC_2.2) [SVID.3]	xdrmem_create(GLIBC_2.2) [SVID.3]	xdrrec_create(GLIBC_2.2) [SVID.3]
xdrrec_eof(GLIBC_2.2) [SVID.3]			

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

__fxstat(GLIBC_2.2) [LSB]	__getpgid(GLIBC_2.2) [LSB]	__lxstat(GLIBC_2.2) [LSB]	__xmknod(GLIBC_2.2) [LSB]
__xstat(GLIBC_2.2) [LSB]	access(GLIBC_2.2) [SUSv3]	acct(GLIBC_2.2) [LSB]	alarm(GLIBC_2.2) [SUSv3]
brk(GLIBC_2.2) [SUSv2]	chdir(GLIBC_2.2) [SUSv3]	chmod(GLIBC_2.2) [SUSv3]	chown(GLIBC_2.2) [SUSv3]
chroot(GLIBC_2.2	clock(GLIBC_2.2)	close(GLIBC_2.2)	closedir(GLIBC_2.

) [SUSv2]	[SUSv3]	[SUSv3]	2) [SUSv3]
creat(GLIBC_2.2) [SUSv3]	dup(GLIBC_2.2) [SUSv3]	dup2(GLIBC_2.2) [SUSv3]	execl(GLIBC_2.2) [SUSv3]
execle(GLIBC_2.2) [SUSv3]	execlp(GLIBC_2.2) [SUSv3]	execv(GLIBC_2.2) [SUSv3]	execve(GLIBC_2.2) [SUSv3]
execvp(GLIBC_2.2) [SUSv3]	exit(GLIBC_2.2) [SUSv3]	fchdir(GLIBC_2.2) [SUSv3]	fchmod(GLIBC_2. 2) [SUSv3]
fchown(GLIBC_2. 2) [SUSv3]	fcntl(GLIBC_2.2) [LSB]	fdatasync(GLIBC_ 2.2) [SUSv3]	flock(GLIBC_2.2) [LSB]
fork(GLIBC_2.2) [SUSv3]	fstatvfs(GLIBC_2. 2) [SUSv3]	fsync(GLIBC_2.2) [SUSv3]	ftime(GLIBC_2.2) [SUSv3]
ftruncate(GLIBC_ 2.2) [SUSv3]	getcontext(GLIBC _2.2) [SUSv3]	getegid(GLIBC_2. 2) [SUSv3]	geteuid(GLIBC_2. 2) [SUSv3]
getgid(GLIBC_2.2) [SUSv3]	getgroups(GLIBC _2.2) [SUSv3]	getitimer(GLIBC_ 2.2) [SUSv3]	getloadavg(GLIB C_2.2) [LSB]
getpagesize(GLIB C_2.2) [SUSv2]	getpgid(GLIBC_2. 2) [SUSv3]	getpgrp(GLIBC_2. 2) [SUSv3]	getpid(GLIBC_2.2) [SUSv3]
getppid(GLIBC_2. 2) [SUSv3]	getpriority(GLIBC _2.2) [SUSv3]	getrlimit(GLIBC_ 2.2) [SUSv3]	getrusage(GLIBC_ 2.2) [SUSv3]
getsid(GLIBC_2.2) [SUSv3]	getuid(GLIBC_2.2) [SUSv3]	getwd(GLIBC_2.2) [SUSv3]	initgroups(GLIBC _2.2) [LSB]
ioctl(GLIBC_2.2) [LSB]	kill(GLIBC_2.2) [LSB]	killpg(GLIBC_2.2) [SUSv3]	lchown(GLIBC_2. 2) [SUSv3]
link(GLIBC_2.2) [LSB]	lockf(GLIBC_2.2) [SUSv3]	lseek(GLIBC_2.2) [SUSv3]	mkdir(GLIBC_2.2) [SUSv3]
mkfifo(GLIBC_2.2) [SUSv3]	mlock(GLIBC_2.2) [SUSv3]	mlockall(GLIBC_2 .2) [SUSv3]	mmap(GLIBC_2.2) [SUSv3]
mprotect(GLIBC_ 2.2) [SUSv3]	msync(GLIBC_2.2) [SUSv3]	munlock(GLIBC_ 2.2) [SUSv3]	munlockall(GLIB C_2.2) [SUSv3]
munmap(GLIBC_ 2.2) [SUSv3]	nanosleep(GLIBC _2.2) [SUSv3]	nice(GLIBC_2.2) [SUSv3]	open(GLIBC_2.2) [SUSv3]
opendir(GLIBC_2. 2) [SUSv3]	pathconf(GLIBC_ 2.2) [SUSv3]	pause(GLIBC_2.2) [SUSv3]	pipe(GLIBC_2.2) [SUSv3]
poll(GLIBC_2.2) [SUSv3]	read(GLIBC_2.2) [SUSv3]	readdir(GLIBC_2. 2) [SUSv3]	readdir_r(GLIBC_ 2.2) [SUSv3]
readlink(GLIBC_2 .2) [SUSv3]	readv(GLIBC_2.2) [SUSv3]	rename(GLIBC_2. 2) [SUSv3]	rmdir(GLIBC_2.2) [SUSv3]
sbrk(GLIBC_2.2) [SUSv2]	sched_get_priorit y_max(GLIBC_2.2) [SUSv3]	sched_get_priorit y_min(GLIBC_2.2) [SUSv3]	sched_getparam(GLIBC_2.2) [SUSv3]
sched_getschedul	sched_rr_get_inte	sched_setparam(	sched_setschedule

er(GLIBC_2.2) [SUSv3]	rval(GLIBC_2.2) [SUSv3]	GLIBC_2.2) [SUSv3]	r(GLIBC_2.2) [SUSv3]
sched_yield(GLIBC_2.2) [SUSv3]	select(GLIBC_2.2) [SUSv3]	setcontext(GLIBC_2.2) [SUSv3]	setegid(GLIBC_2.2) [SUSv3]
seteuid(GLIBC_2.2) [SUSv3]	setgid(GLIBC_2.2) [SUSv3]	setitimer(GLIBC_2.2) [SUSv3]	setpgid(GLIBC_2.2) [SUSv3]
setpgrp(GLIBC_2.2) [SUSv3]	setpriority(GLIBC_2.2) [SUSv3]	setregid(GLIBC_2.2) [SUSv3]	setreuid(GLIBC_2.2) [SUSv3]
setrlimit(GLIBC_2.2) [SUSv3]	setrlimit64(GLIBC_2.2) [LFS]	setsid(GLIBC_2.2) [SUSv3]	setuid(GLIBC_2.2) [SUSv3]
sleep(GLIBC_2.2) [SUSv3]	statvfs(GLIBC_2.2) [SUSv3]	stime(GLIBC_2.2) [LSB]	symlink(GLIBC_2.2) [SUSv3]
sync(GLIBC_2.2) [SUSv3]	sysconf(GLIBC_2.2) [SUSv3]	time(GLIBC_2.2) [SUSv3]	times(GLIBC_2.2) [SUSv3]
truncate(GLIBC_2.2) [SUSv3]	ulimit(GLIBC_2.2) [SUSv3]	umask(GLIBC_2.2) [SUSv3]	uname(GLIBC_2.2) [SUSv3]
unlink(GLIBC_2.2) [LSB]	utime(GLIBC_2.2) [SUSv3]	utimes(GLIBC_2.2) [SUSv3]	vfork(GLIBC_2.2) [SUSv3]
wait(GLIBC_2.2) [SUSv3]	wait4(GLIBC_2.2) [LSB]	waitpid(GLIBC_2.2) [LSB]	write(GLIBC_2.2) [SUSv3]
writev(GLIBC_2.2) [SUSv3]			

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

_IO_feof(GLIBC_2.2) [LSB]	_IO_getc(GLIBC_2.2) [LSB]	_IO_putc(GLIBC_2.2) [LSB]	_IO_puts(GLIBC_2.2) [LSB]
asprintf(GLIBC_2.2) [LSB]	clearerr(GLIBC_2.2) [SUSv3]	ctermid(GLIBC_2.2) [SUSv3]	fclose(GLIBC_2.2) [SUSv3]
fdopen(GLIBC_2.2) [SUSv3]	feof(GLIBC_2.2) [SUSv3]	ferror(GLIBC_2.2) [SUSv3]	fflush(GLIBC_2.2) [SUSv3]
fflush_unlocked(GLIBC_2.2) [LSB]	fgetc(GLIBC_2.2) [SUSv3]	fgetpos(GLIBC_2.2) [SUSv3]	fgets(GLIBC_2.2) [SUSv3]
fgetwc_unlocked(GLIBC_2.2) [LSB]	fileno(GLIBC_2.2) [SUSv3]	flockfile(GLIBC_2.2) [SUSv3]	fopen(GLIBC_2.2) [SUSv3]
fprintf(GLIBC_2.2) [SUSv3]	fputc(GLIBC_2.2) [SUSv3]	fputs(GLIBC_2.2) [SUSv3]	fread(GLIBC_2.2) [SUSv3]

freopen(GLIBC_2.2) [SUSv3]	fscanf(GLIBC_2.2) [LSB]	fseek(GLIBC_2.2) [SUSv3]	fseeko(GLIBC_2.2) [SUSv3]
fsetpos(GLIBC_2.2) [SUSv3]	ftell(GLIBC_2.2) [SUSv3]	ftello(GLIBC_2.2) [SUSv3]	fwrite(GLIBC_2.2) [SUSv3]
getc(GLIBC_2.2) [SUSv3]	getc_unlocked(GLIBC_2.2) [SUSv3]	getchar(GLIBC_2.2) [SUSv3]	getchar_unlocked (GLIBC_2.2) [SUSv3]
getw(GLIBC_2.2) [SUSv2]	pclose(GLIBC_2.2) [SUSv3]	popen(GLIBC_2.2) [SUSv3]	printf(GLIBC_2.2) [SUSv3]
putc(GLIBC_2.2) [SUSv3]	putc_unlocked(GLIBC_2.2) [SUSv3]	putchar(GLIBC_2.2) [SUSv3]	putchar_unlocked (GLIBC_2.2) [SUSv3]
puts(GLIBC_2.2) [SUSv3]	putw(GLIBC_2.2) [SUSv2]	remove(GLIBC_2.2) [SUSv3]	rewind(GLIBC_2.2) [SUSv3]
rewinddir(GLIBC_2.2) [SUSv3]	scanf(GLIBC_2.2) [LSB]	seekdir(GLIBC_2.2) [SUSv3]	setbuf(GLIBC_2.2) [SUSv3]
setbuffer(GLIBC_2.2) [LSB]	setvbuf(GLIBC_2.2) [SUSv3]	snprintf(GLIBC_2.2) [SUSv3]	sprintf(GLIBC_2.2) [SUSv3]
sscanf(GLIBC_2.2) [LSB]	telldir(GLIBC_2.2) [SUSv3]	tempnam(GLIBC_2.2) [SUSv3]	ungetc(GLIBC_2.2) [SUSv3]
vasprintf(GLIBC_2.2) [LSB]	vdprintf(GLIBC_2.2) [LSB]	vfprintf(GLIBC_2.2) [SUSv3]	vprintf(GLIBC_2.2) [SUSv3]
vsnprintf(GLIBC_2.2) [SUSv3]	vsprintf(GLIBC_2.2) [SUSv3]		

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

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

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

__libc_current_sigrtmax(GLIBC_2.2) [LSB]	__libc_current_sigrtmin(GLIBC_2.2) [LSB]	__sigsetjmp(GLIBC_2.2) [LSB]	__sysv_signal(GLIBC_2.2) [LSB]

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

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

_sys_siglist(GLIBC_2.3.3) [LSB]			

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

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

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

_nl_msg_cat_cntr(GLIBC_2.2) [LSB]			

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

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

11.2.7 Wide Characters

11.2.7.1 Interfaces for Wide Characters

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

Table 11-11 libc - Wide Characters Function Interfaces

__wcstod_internal(GLIBC_2.2) [LSB]	__wcstof_internal(GLIBC_2.2) [LSB]	__wcstol_internal(GLIBC_2.2) [LSB]	__wcstold_internal(GLIBC_2.2) [LSB]
__wcstoul_internal(GLIBC_2.2)	btowc(GLIBC_2.2) [SUSv3]	fgetwc(GLIBC_2.2) [SUSv3]	fgetws(GLIBC_2.2) [SUSv3]

[LSB]			
fputwc(GLIBC_2.2) [SUSv3]	fputws(GLIBC_2.2) [SUSv3]	fwide(GLIBC_2.2) [SUSv3]	fwprintf(GLIBC_2.2) [SUSv3]
fwscanf(GLIBC_2.2) [LSB]	getwc(GLIBC_2.2) [SUSv3]	getwchar(GLIBC_2.2) [SUSv3]	mblen(GLIBC_2.2) [SUSv3]
mbrlen(GLIBC_2.2) [SUSv3]	mbrtowc(GLIBC_2.2) [SUSv3]	mbsinit(GLIBC_2.2) [SUSv3]	mbsnrtowcs(GLIBC_2.2) [LSB]
mbsrtowcs(GLIBC_2.2) [SUSv3]	mbstowcs(GLIBC_2.2) [SUSv3]	mbtowc(GLIBC_2.2) [SUSv3]	putwc(GLIBC_2.2) [SUSv3]
putwchar(GLIBC_2.2) [SUSv3]	swprintf(GLIBC_2.2) [SUSv3]	swscanf(GLIBC_2.2) [LSB]	towctrans(GLIBC_2.2) [SUSv3]
towlower(GLIBC_2.2) [SUSv3]	towupper(GLIBC_2.2) [SUSv3]	ungetwc(GLIBC_2.2) [SUSv3]	vfwprintf(GLIBC_2.2) [SUSv3]
vfwscanf(GLIBC_2.2) [LSB]	vswprintf(GLIBC_2.2) [SUSv3]	vswscanf(GLIBC_2.2) [LSB]	vwprintf(GLIBC_2.2) [SUSv3]
vwscanf(GLIBC_2.2) [LSB]	wcpcpy(GLIBC_2.2) [LSB]	wcpncpy(GLIBC_2.2) [LSB]	wcrtomb(GLIBC_2.2) [SUSv3]
wcscasecmp(GLIBC_2.2) [LSB]	wcscat(GLIBC_2.2) [SUSv3]	wcschr(GLIBC_2.2) [SUSv3]	wcscmp(GLIBC_2.2) [SUSv3]
wcscoll(GLIBC_2.2) [SUSv3]	wcscpy(GLIBC_2.2) [SUSv3]	wcscspn(GLIBC_2.2) [SUSv3]	wcsdup(GLIBC_2.2) [LSB]
wcsftime(GLIBC_2.2) [SUSv3]	wcslen(GLIBC_2.2) [SUSv3]	wcsncasecmp(GLIBC_2.2) [LSB]	wcsncat(GLIBC_2.2) [SUSv3]
wcsncmp(GLIBC_2.2) [SUSv3]	wcsncpy(GLIBC_2.2) [SUSv3]	wcsnlen(GLIBC_2.2) [LSB]	wcsnrtombs(GLIBC_2.2) [LSB]
wcspbrk(GLIBC_2.2) [SUSv3]	wcsrchr(GLIBC_2.2) [SUSv3]	wcsrtombs(GLIBC_2.2) [SUSv3]	wcsspn(GLIBC_2.2) [SUSv3]
wcsstr(GLIBC_2.2) [SUSv3]	wcstod(GLIBC_2.2) [SUSv3]	wcstof(GLIBC_2.2) [SUSv3]	wcstoimax(GLIBC_2.2) [SUSv3]
wcstok(GLIBC_2.2) [SUSv3]	wcstol(GLIBC_2.2) [SUSv3]	wcstold(GLIBC_2.2) [SUSv3]	wcstoll(GLIBC_2.2) [SUSv3]
wcstombs(GLIBC_2.2) [SUSv3]	wcstoq(GLIBC_2.2) [LSB]	wcstoul(GLIBC_2.2) [SUSv3]	wcstoull(GLIBC_2.2) [SUSv3]
wcstoumax(GLIBC_2.2) [SUSv3]	wcstouq(GLIBC_2.2) [LSB]	wcswcs(GLIBC_2.2) [SUSv3]	wcswidth(GLIBC_2.2) [SUSv3]
wcsxfrm(GLIBC_2.2) [SUSv3]	wctob(GLIBC_2.2) [SUSv3]	wctomb(GLIBC_2.2) [SUSv3]	wctrans(GLIBC_2.2) [SUSv3]
wctype(GLIBC_2.2) [SUSv3]	wcwidth(GLIBC_2.2) [SUSv3]	wmemchr(GLIBC_2.2) [SUSv3]	wmemcmp(GLIBC_2.2) [SUSv3]
wmemcpy(GLIBC_2.2) [SUSv3]	wmemmove(GLIBC_2.2) [SUSv3]	wmemset(GLIBC_2.2) [SUSv3]	wprintf(GLIBC_2.2) [SUSv3]

wscanf(GLIBC_2.2) [LSB]			

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

__mempcpy(GLIBC_2.2) [LSB]	__rawmemchr(GLIBC_2.2) [LSB]	__stpcpy(GLIBC_2.2) [LSB]	__strdup(GLIBC_2.2) [LSB]
__strtod_internal(GLIBC_2.2) [LSB]	__strtof_internal(GLIBC_2.2) [LSB]	__strtok_r(GLIBC_2.2) [LSB]	__strtol_internal(GLIBC_2.2) [LSB]
__strtold_internal(GLIBC_2.2) [LSB]	__strtoll_internal(GLIBC_2.2) [LSB]	__strtoul_internal(GLIBC_2.2) [LSB]	__strtoull_internal(GLIBC_2.2) [LSB]
bcmp(GLIBC_2.2) [SUSv3]	bcopy(GLIBC_2.2) [SUSv3]	bzero(GLIBC_2.2) [SUSv3]	ffs(GLIBC_2.2) [SUSv3]
index(GLIBC_2.2) [SUSv3]	memccpy(GLIBC_2.2) [SUSv3]	memchr(GLIBC_2.2) [SUSv3]	memcmp(GLIBC_2.2) [SUSv3]
memcpy(GLIBC_2.2) [SUSv3]	memmove(GLIBC_2.2) [SUSv3]	memrchr(GLIBC_2.2) [LSB]	memset(GLIBC_2.2) [SUSv3]
rindex(GLIBC_2.2) [SUSv3]	stpcpy(GLIBC_2.2) [LSB]	stpncpy(GLIBC_2.2) [LSB]	strcasecmp(GLIBC_2.2) [SUSv3]
strcasestr(GLIBC_2.2) [LSB]	strcat(GLIBC_2.2) [SUSv3]	strchr(GLIBC_2.2) [SUSv3]	strcmp(GLIBC_2.2) [SUSv3]
strcoll(GLIBC_2.2) [SUSv3]	strcpy(GLIBC_2.2) [SUSv3]	strcspn(GLIBC_2.2) [SUSv3]	strdup(GLIBC_2.2) [SUSv3]
strerror(GLIBC_2.2) [SUSv3]	strerror_r(GLIBC_2.2) [LSB]	strfmon(GLIBC_2.2) [SUSv3]	strftime(GLIBC_2.2) [SUSv3]
strlen(GLIBC_2.2) [SUSv3]	strncasecmp(GLIBC_2.2) [SUSv3]	strncat(GLIBC_2.2) [SUSv3]	strncmp(GLIBC_2.2) [SUSv3]
strncpy(GLIBC_2.2) [SUSv3]	strndup(GLIBC_2.2) [LSB]	strnlen(GLIBC_2.2) [LSB]	strpbrk(GLIBC_2.2) [SUSv3]
strptime(GLIBC_2.2) [LSB]	strrchr(GLIBC_2.2) [SUSv3]	strsep(GLIBC_2.2) [LSB]	strsignal(GLIBC_2.2) [LSB]
strspn(GLIBC_2.2) [SUSv3]	strstr(GLIBC_2.2) [SUSv3]	strtof(GLIBC_2.2) [SUSv3]	strtoimax(GLIBC_2.2) [SUSv3]
strtok(GLIBC_2.2) [SUSv3]	strtok_r(GLIBC_2.2) [SUSv3]	strtold(GLIBC_2.2) [SUSv3]	strtoll(GLIBC_2.2) [SUSv3]
strtoq(GLIBC_2.2) [LSB]	strtoull(GLIBC_2.2) [SUSv3]	strtoumax(GLIBC_2.2) [SUSv3]	strtouq(GLIBC_2.2) [LSB]

strxfrm(GLIBC_2.2) [SUSv3]	swab(GLIBC_2.2) [SUSv3]		

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 11-13 libc - IPC Functions Function Interfaces

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

11.2.10 Regular Expressions

11.2.10.1 Interfaces for Regular Expressions

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

Table 11-14 libc - Regular Expressions Function Interfaces

regcomp(GLIBC_2.2) [SUSv3]	regerror(GLIBC_2.2) [SUSv3]	regexec(GLIBC_2.3.4) [LSB]	regfree(GLIBC_2.2) [SUSv3]

11.2.11 Character Type Functions

11.2.11.1 Interfaces for Character Type Functions

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

Table 11-15 libc - Character Type Functions Function Interfaces

__ctype_get_mb_cur_max(GLIBC_2.2) [LSB]	_tolower(GLIBC_2.2) [SUSv3]	_toupper(GLIBC_2.2) [SUSv3]	isalnum(GLIBC_2.2) [SUSv3]
isalpha(GLIBC_2.2) [SUSv3]	isascii(GLIBC_2.2) [SUSv3]	iscntrl(GLIBC_2.2) [SUSv3]	isdigit(GLIBC_2.2) [SUSv3]
isgraph(GLIBC_2.2) [SUSv3]	islower(GLIBC_2.2) [SUSv3]	isprint(GLIBC_2.2) [SUSv3]	ispunct(GLIBC_2.2) [SUSv3]
isspace(GLIBC_2.2) [SUSv3]	isupper(GLIBC_2.2) [SUSv3]	iswalnum(GLIBC_2.2) [SUSv3]	iswalpha(GLIBC_2.2) [SUSv3]

iswblank(GLIBC_2.2) [SUSv3]	iswcntrl(GLIBC_2.2) [SUSv3]	iswctype(GLIBC_2.2) [SUSv3]	iswdigit(GLIBC_2.2) [SUSv3]
iswgraph(GLIBC_2.2) [SUSv3]	iswlower(GLIBC_2.2) [SUSv3]	iswprint(GLIBC_2.2) [SUSv3]	iswpunct(GLIBC_2.2) [SUSv3]
iswspace(GLIBC_2.2) [SUSv3]	iswupper(GLIBC_2.2) [SUSv3]	iswxdigit(GLIBC_2.2) [SUSv3]	isxdigit(GLIBC_2.2) [SUSv3]
toascii(GLIBC_2.2) [SUSv3]	tolower(GLIBC_2.2) [SUSv3]	toupper(GLIBC_2.2) [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

adjtime(GLIBC_2.2) [LSB]	asctime(GLIBC_2.2) [SUSv3]	asctime_r(GLIBC_2.2) [SUSv3]	ctime(GLIBC_2.2) [SUSv3]
ctime_r(GLIBC_2.2) [SUSv3]	difftime(GLIBC_2.2) [SUSv3]	gmtime(GLIBC_2.2) [SUSv3]	gmtime_r(GLIBC_2.2) [SUSv3]
localtime(GLIBC_2.2) [SUSv3]	localtime_r(GLIBC_2.2) [SUSv3]	mktime(GLIBC_2.2) [SUSv3]	tzset(GLIBC_2.2) [SUSv3]
ualarm(GLIBC_2.2) [SUSv3]			

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

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

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

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

C_2.2) [SUSv3]	C_2.2) [SUSv3]	C_2.2) [LSB]	C_2.2) [SUSv3]
cfsetospeed(GLIBC_2.2) [SUSv3]	cfsetspeed(GLIBC_2.2) [LSB]	tcdrain(GLIBC_2.2) [SUSv3]	tcflow(GLIBC_2.2) [SUSv3]
tcflush(GLIBC_2.2) [SUSv3]	tcgetattr(GLIBC_2.2) [SUSv3]	tcgetpgrp(GLIBC_2.2) [SUSv3]	tcgetsid(GLIBC_2.2) [SUSv3]
tcsendbreak(GLIBC_2.2) [SUSv3]	tcsetattr(GLIBC_2.2) [SUSv3]	tcsetpgrp(GLIBC_2.2) [SUSv3]	

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

endgrent(GLIBC_2.2) [SUSv3]	endprotoent(GLIBC_2.2) [SUSv3]	endpwent(GLIBC_2.2) [SUSv3]	endservent(GLIBC_2.2) [SUSv3]
endutent(GLIBC_2.2) [SUSv2]	endutxent(GLIBC_2.2) [SUSv3]	getgrent(GLIBC_2.2) [SUSv3]	getgrgid(GLIBC_2.2) [SUSv3]
getgrgid_r(GLIBC_2.2) [SUSv3]	getgrnam(GLIBC_2.2) [SUSv3]	getgrnam_r(GLIBC_2.2) [SUSv3]	getgrouplist(GLIBC_2.2.4) [LSB]
gethostbyaddr(GLIBC_2.2) [SUSv3]	gethostbyname(GLIBC_2.2) [SUSv3]	getprotobyname(GLIBC_2.2) [SUSv3]	getprotobynumber(GLIBC_2.2) [SUSv3]
getprotoent(GLIBC_2.2) [SUSv3]	getpwent(GLIBC_2.2) [SUSv3]	getpwnam(GLIBC_2.2) [SUSv3]	getpwnam_r(GLIBC_2.2) [SUSv3]
getpwuid(GLIBC_2.2) [SUSv3]	getpwuid_r(GLIBC_2.2) [SUSv3]	getservbyname(GLIBC_2.2) [SUSv3]	getservbyport(GLIBC_2.2) [SUSv3]
getservent(GLIBC_2.2) [SUSv3]	getutent(GLIBC_2.2) [LSB]	getutent_r(GLIBC_2.2) [LSB]	getutxent(GLIBC_2.2) [SUSv3]
getutxid(GLIBC_2.2) [SUSv3]	getutxline(GLIBC_2.2) [SUSv3]	pututxline(GLIBC_2.2) [SUSv3]	setgrent(GLIBC_2.2) [SUSv3]
setgroups(GLIBC_2.2) [LSB]	setprotoent(GLIBC_2.2) [SUSv3]	setpwent(GLIBC_2.2) [SUSv3]	setservent(GLIBC_2.2) [SUSv3]
setutent(GLIBC_2.2) [LSB]	setutxent(GLIBC_2.2) [SUSv3]	utmpname(GLIBC_2.2) [LSB]	

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.

122 **Table 11-20 libc - Language Support Function Interfaces**

__libc_start_main(GLIBC_2.2) [LSB]			

123

11.2.16 Large File Support

124 ### 11.2.16.1 Interfaces for Large File Support

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

128 **Table 11-21 libc - Large File Support Function Interfaces**

__fxstat64(GLIBC _2.2) [LSB]	__lxstat64(GLIBC _2.2) [LSB]	__xstat64(GLIBC_ 2.2) [LSB]	creat64(GLIBC_2. 2) [LFS]
fgetpos64(GLIBC_ 2.2) [LFS]	fopen64(GLIBC_2. 2) [LFS]	freopen64(GLIBC _2.2) [LFS]	fseeko64(GLIBC_2 .2) [LFS]
fsetpos64(GLIBC_ 2.2) [LFS]	fstatvfs64(GLIBC_ 2.2) [LFS]	ftello64(GLIBC_2. 2) [LFS]	ftruncate64(GLIB C_2.2) [LFS]
ftw64(GLIBC_2.2) [LFS]	getrlimit64(GLIB C_2.2) [LFS]	lockf64(GLIBC_2. 2) [LFS]	mkstemp64(GLIB C_2.2) [LFS]
mmap64(GLIBC_ 2.2) [LFS]	nftw64(GLIBC_2.3 .3) [LFS]	readdir64(GLIBC_ 2.2) [LFS]	statvfs64(GLIBC_ 2.2) [LFS]
tmpfile64(GLIBC_ 2.2) [LFS]	truncate64(GLIBC _2.2) [LFS]		

129

11.2.17 Standard Library

130 ### 11.2.17.1 Interfaces for Standard Library

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

134 **Table 11-22 libc - Standard Library Function Interfaces**

_Exit(GLIBC_2.2) [SUSv3]	__assert_fail(GLIB C_2.2) [LSB]	__cxa_atexit(GLIB C_2.2) [LSB]	__errno_location(GLIBC_2.2) [LSB]
__fpending(GLIB C_2.2) [LSB]	__getpagesize(GL IBC_2.2) [LSB]	__isinf(GLIBC_2.2) [LSB]	__isinff(GLIBC_2. 2) [LSB]
__isinfl(GLIBC_2. 2) [LSB]	__isnan(GLIBC_2. 2) [LSB]	__isnanf(GLIBC_2 .2) [LSB]	__isnanl(GLIBC_2 .2) [LSB]
__sysconf(GLIBC_ 2.2) [LSB]	_exit(GLIBC_2.2) [SUSv3]	_longjmp(GLIBC_ 2.2) [SUSv3]	_setjmp(GLIBC_2. 2) [SUSv3]
a64l(GLIBC_2.2) [SUSv3]	abort(GLIBC_2.2) [SUSv3]	abs(GLIBC_2.2) [SUSv3]	atof(GLIBC_2.2) [SUSv3]
atoi(GLIBC_2.2)	atol(GLIBC_2.2)	atoll(GLIBC_2.2)	basename(GLIBC

[SUSv3]	[SUSv3]	[SUSv3]	_2.2) [SUSv3]
bsearch(GLIBC_2.2) [SUSv3]	calloc(GLIBC_2.2) [SUSv3]	closelog(GLIBC_2.2) [SUSv3]	confstr(GLIBC_2.2) [SUSv3]
cuserid(GLIBC_2.2) [SUSv2]	daemon(GLIBC_2.2) [LSB]	dirname(GLIBC_2.2) [SUSv3]	div(GLIBC_2.2) [SUSv3]
drand48(GLIBC_2.2) [SUSv3]	ecvt(GLIBC_2.2) [SUSv3]	erand48(GLIBC_2.2) [SUSv3]	err(GLIBC_2.2) [LSB]
error(GLIBC_2.2) [LSB]	errx(GLIBC_2.2) [LSB]	fcvt(GLIBC_2.2) [SUSv3]	fmtmsg(GLIBC_2.2) [SUSv3]
fnmatch(GLIBC_2.2.3) [SUSv3]	fpathconf(GLIBC_2.2) [SUSv3]	free(GLIBC_2.2) [SUSv3]	freeaddrinfo(GLIBC_2.2) [SUSv3]
ftrylockfile(GLIBC_2.2) [SUSv3]	ftw(GLIBC_2.2) [SUSv3]	funlockfile(GLIBC_2.2) [SUSv3]	gai_strerror(GLIBC_2.2) [SUSv3]
gcvt(GLIBC_2.2) [SUSv3]	getaddrinfo(GLIBC_2.2) [SUSv3]	getcwd(GLIBC_2.2) [SUSv3]	getdate(GLIBC_2.2) [SUSv3]
getenv(GLIBC_2.2) [SUSv3]	getlogin(GLIBC_2.2) [SUSv3]	getlogin_r(GLIBC_2.2) [SUSv3]	getnameinfo(GLIBC_2.2) [SUSv3]
getopt(GLIBC_2.2) [LSB]	getopt_long(GLIBC_2.2) [LSB]	getopt_long_only(GLIBC_2.2) [LSB]	getsubopt(GLIBC_2.2) [SUSv3]
gettimeofday(GLIBC_2.2) [SUSv3]	glob(GLIBC_2.2) [SUSv3]	glob64(GLIBC_2.2) [LSB]	globfree(GLIBC_2.2) [SUSv3]
globfree64(GLIBC_2.2) [LSB]	grantpt(GLIBC_2.2) [SUSv3]	hcreate(GLIBC_2.2) [SUSv3]	hdestroy(GLIBC_2.2) [SUSv3]
hsearch(GLIBC_2.2) [SUSv3]	htonl(GLIBC_2.2) [SUSv3]	htons(GLIBC_2.2) [SUSv3]	imaxabs(GLIBC_2.2) [SUSv3]
imaxdiv(GLIBC_2.2) [SUSv3]	inet_addr(GLIBC_2.2) [SUSv3]	inet_ntoa(GLIBC_2.2) [SUSv3]	inet_ntop(GLIBC_2.2) [SUSv3]
inet_pton(GLIBC_2.2) [SUSv3]	initstate(GLIBC_2.2) [SUSv3]	insque(GLIBC_2.2) [SUSv3]	isatty(GLIBC_2.2) [SUSv3]
isblank(GLIBC_2.2) [SUSv3]	jrand48(GLIBC_2.2) [SUSv3]	l64a(GLIBC_2.2) [SUSv3]	labs(GLIBC_2.2) [SUSv3]
lcong48(GLIBC_2.2) [SUSv3]	ldiv(GLIBC_2.2) [SUSv3]	lfind(GLIBC_2.2) [SUSv3]	llabs(GLIBC_2.2) [SUSv3]
lldiv(GLIBC_2.2) [SUSv3]	longjmp(GLIBC_2.2) [SUSv3]	lrand48(GLIBC_2.2) [SUSv3]	lsearch(GLIBC_2.2) [SUSv3]
makecontext(GLIBC_2.2) [SUSv3]	malloc(GLIBC_2.2) [SUSv3]	memmem(GLIBC_2.2) [LSB]	mkstemp(GLIBC_2.2) [SUSv3]
mktemp(GLIBC_2.2) [SUSv3]	mrand48(GLIBC_2.2) [SUSv3]	nftw(GLIBC_2.3.3) [SUSv3]	nrand48(GLIBC_2.2) [SUSv3]
ntohl(GLIBC_2.2) [SUSv3]	ntohs(GLIBC_2.2) [SUSv3]	openlog(GLIBC_2.2) [SUSv3]	perror(GLIBC_2.2) [SUSv3]

posix_memalign(GLIBC_2.2) [SUSv3]	posix_openpt(GLIBC_2.2.1) [SUSv3]	ptsname(GLIBC_2.2) [SUSv3]	putenv(GLIBC_2.2) [SUSv3]
qsort(GLIBC_2.2) [SUSv3]	rand(GLIBC_2.2) [SUSv3]	rand_r(GLIBC_2.2) [SUSv3]	random(GLIBC_2.2) [SUSv3]
realloc(GLIBC_2.2) [SUSv3]	realpath(GLIBC_2.3) [SUSv3]	remque(GLIBC_2.2) [SUSv3]	seed48(GLIBC_2.2) [SUSv3]
setenv(GLIBC_2.2) [SUSv3]	sethostname(GLIBC_2.2) [LSB]	setlogmask(GLIBC_2.2) [SUSv3]	setstate(GLIBC_2.2) [SUSv3]
srand(GLIBC_2.2) [SUSv3]	srand48(GLIBC_2.2) [SUSv3]	srandom(GLIBC_2.2) [SUSv3]	strtod(GLIBC_2.2) [SUSv3]
strtol(GLIBC_2.2) [SUSv3]	strtoul(GLIBC_2.2) [SUSv3]	swapcontext(GLIBC_2.2) [SUSv3]	syslog(GLIBC_2.2) [SUSv3]
system(GLIBC_2.2) [LSB]	tdelete(GLIBC_2.2) [SUSv3]	tfind(GLIBC_2.2) [SUSv3]	tmpfile(GLIBC_2.2) [SUSv3]
tmpnam(GLIBC_2.2) [SUSv3]	tsearch(GLIBC_2.2) [SUSv3]	ttyname(GLIBC_2.2) [SUSv3]	ttyname_r(GLIBC_2.2) [SUSv3]
twalk(GLIBC_2.2) [SUSv3]	unlockpt(GLIBC_2.2) [SUSv3]	unsetenv(GLIBC_2.2) [SUSv3]	usleep(GLIBC_2.2) [SUSv3]
verrx(GLIBC_2.2) [LSB]	vfscanf(GLIBC_2.2) [LSB]	vscanf(GLIBC_2.2) [LSB]	vsscanf(GLIBC_2.2) [LSB]
vsyslog(GLIBC_2.2) [LSB]	warn(GLIBC_2.2) [LSB]	warnx(GLIBC_2.2) [LSB]	wordexp(GLIBC_2.2) [SUSv3]
wordfree(GLIBC_2.2) [SUSv3]			

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

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

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       35

extern int *__errno_location(void);
```

11.3.7 fcntl.h

```

#define F_GETLK64       5
#define F_SETLK64       6
#define F_SETLKW64      7

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 int intmax_t;
typedef unsigned long int uintmax_t;
typedef unsigned long int uintptr_t;
typedef unsigned 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 uintmax_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 ULONG_MAX       0xFFFFFFFFFFFFFFFFUL
#define LONG_MAX        9223372036854775807L

#define CHAR_MIN        0
#define CHAR_MAX        255

#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 struct if_nameindex *if_nameindex(void);
extern unsigned int if_nametoindex(const char *);
```

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 *getpwuid(uid_t);
extern void setpwent(void);
extern int getpwnam_r(char *, struct passwd *, char *, size_t,
                      struct passwd **);
extern int getpwuid_r(uid_t, 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_perrno(enum clnt_stat);
extern void clnt_perror(struct CLIENT *, const char *);
extern char *clnt_spcreateerror(const char *);
extern char *clnt_sperrno(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/rpc_msg.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 *);
```

```
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 xdr_opaque(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)
    );
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_getscheduler(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);
```

11.3.41 search.h

```
extern int hcreate(size_t);
extern ENTRY *hsearch(ENTRY, ACTION);
extern void insque(void *, void *);
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 long int __jmp_buf[18];

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 __NUM_ACRS      16
#define __NUM_FPRS      16
#define __NUM_GPRS      16

typedef struct {
    unsigned long int mask;
    unsigned long int addr;
} __attribute__ ((aligned(8)))
    _psw_t;
typedef struct {
    _psw_t psw;
    unsigned long int gprs[16];
    unsigned int acrs[16];
} _s390_regs_common;

#define SIGEV_PAD_SIZE  ((SIGEV_MAX_SIZE/sizeof(int))-4)

#define SI_PAD_SIZE     ((SI_MAX_SIZE/sizeof(int))-4)

struct sigaction {
    union {
        sighandler_t _sa_handler;
        void (*_sa_sigaction) (int, siginfo_t *, void *);
    } __sigaction_handler;
    unsigned long int sa_flags;
    void (*sa_restorer) (void);
    sigset_t sa_mask;
};

#define MINSIGSTKSZ     2048
#define SIGSTKSZ        8192
```

```
typedef struct {
    unsigned int fpc;
    double fprs[__NUM_FPRS];
} _s390_fp_regs;
typedef struct {
    _s390_regs_common regs;
    _s390_fp_regs fpregs;
} _sigregs;

struct sigcontext {
    unsigned long int oldmask;
    _sigregs *sregs;
};
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 sigisemptyset(const sigset_t *);
extern int sigismember(const sigset_t *, int);
extern int sigorset(sigset_t *, const sigset_t *, const sigset_t *);
extern int sigpending(sigset_t *);
extern int sigrelse(int);
extern sighandler_t sigset(int, sighandler_t);
extern int pthread_kill(pthread_t, int);
extern int pthread_sigmask(int, sigset_t *, sigset_t *);
extern int sigaction(int, const struct sigaction *, struct sigaction *);
extern int sigwait(sigset_t *, int *);
extern int kill(pid_t, int);
extern int sigaltstack(const struct sigaltstack *, struct sigaltstack
*);
extern sighandler_t signal(int, sighandler_t);
extern int sigpause(int);
extern int sigprocmask(int, const sigset_t *, sigset_t *);
extern int sigreturn(struct sigcontext *);
extern int sigsuspend(const sigset_t *);
extern int sigqueue(pid_t, int, const union sigval);
extern int sigwaitinfo(const sigset_t *, siginfo_t *);
extern int sigtimedwait(const sigset_t *, siginfo_t *,
                        const struct timespec *);
extern sighandler_t bsd_signal(int, sighandler_t);
```

11.3.44 stddef.h

```

typedef unsigned long int size_t;
typedef long int ptrdiff_t;
```

11.3.45 stdio.h

```

#define __IO_FILE_SIZE  216

extern char *const _sys_errlist(void);
```

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

```
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 __strtoul_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, int *, int *);
extern char *gcvt(double, int, char *);
extern char *getenv(const char *);
extern int getsubopt(char **, char *const *, char **);
extern int grantpt(int);
extern long int jrand48(unsigned short);
extern char *l64a(long int);
extern long int labs(long int);
extern void lcong48(unsigned short);
extern ldiv_t ldiv(long int, long int);
extern long long int llabs(long long int);
extern lldiv_t lldiv(long long int, long long int);
extern long int lrand48(void);
extern 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 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 short *seed48(unsigned short);
```

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

11.3.47 string.h

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

```
extern size_t strxfrm(char *, const char *, size_t);
extern int bcmp(void *, void *, size_t);
extern void bzero(void *, size_t);
extern int ffs(int);
extern char *index(char *, int);
extern void *memccpy(void *, const void *, int, size_t);
extern char *rindex(char *, int);
extern int strcasecmp(char *, char *);
extern char *strdup(char *);
extern int strncasecmp(char *, char *, size_t);
extern char *strndup(const char *, size_t);
extern size_t strnlen(const char *, size_t);
extern char *strsep(char **, const char *);
extern char *strerror_r(int, char *, size_t);
extern char *strtok_r(char *, const char *, char **);
extern char *strcasestr(const char *, const char *);
extern char *stpcpy(char *, const char *);
extern char *stpncpy(char *, 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
#define FIONREAD        21531
#define TIOCNOTTY       21538

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;
    mode_t mode;
    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;
    time_t msg_rtime;
    time_t msg_ctime;
    unsigned long int __msg_cbytes;
    msgqnum_t msg_qnum;
    msglen_t msg_qbytes;
    pid_t msg_lspid;
    pid_t msg_lrpid;
    unsigned long int __unused4;
    unsigned long int __unused5;
};
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;
    time_t sem_ctime;
    unsigned long int sem_nsems;
    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  4096

typedef unsigned long int shmatt_t;

struct shmid_ds {
    struct ipc_perm shm_perm;
    size_t shm_segsz;
    time_t shm_atime;
    time_t shm_dtime;
    time_t shm_ctime;
    pid_t shm_cpid;
    pid_t shm_lpid;
    shmatt_t shm_nattch;
    unsigned long int __unused4;
    unsigned long int __unused5;
};
extern int __getpagesize(void);
extern void *shmat(int, const void *, int);
extern int shmctl(int, int, struct shmid_ds *);
extern int shmdt(const void *);
extern int shmget(key_t, size_t, int);
```

11.3.58 sys/socket.h

```

typedef uint64_t __ss_aligntype;

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

extern int bind(int, const struct sockaddr *, socklen_t);
extern int getnameinfo(const struct sockaddr *, socklen_t, char *,
                       socklen_t, char *, socklen_t, unsigned int);
extern int getsockname(int, struct sockaddr *, socklen_t *);
extern int listen(int, int);
extern int setsockopt(int, int, int, const void *, socklen_t);
extern int accept(int, struct sockaddr *, socklen_t *);
extern int connect(int, const struct sockaddr *, socklen_t);
extern ssize_t recv(int, void *, size_t, int);
extern ssize_t recvfrom(int, void *, size_t, int, struct sockaddr *,
                        socklen_t *);
extern ssize_t recvmsg(int, struct msghdr *, int);
extern ssize_t send(int, const void *, size_t, int);
extern ssize_t sendmsg(int, const struct msghdr *, int);
extern ssize_t sendto(int, const void *, size_t, int,
                      const struct sockaddr *, socklen_t);
extern int getpeername(int, struct sockaddr *, socklen_t *);
extern int getsockopt(int, int, int, void *, socklen_t *);
extern int shutdown(int, int);
```

```
extern int socket(int, int, int);
extern int socketpair(int, int, int, int);
extern int sockatmark(int);
```

11.3.59 sys/stat.h

```

#define _STAT_VER      1

struct stat {
    dev_t st_dev;
    ino_t st_ino;
    nlink_t st_nlink;
    mode_t st_mode;
    uid_t st_uid;
    gid_t st_gid;
    int pad0;
    dev_t st_rdev;
    off_t st_size;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    blksize_t st_blksize;
    blkcnt_t st_blocks;
    long int __unused[3];
};
struct stat64 {
    dev_t st_dev;
    ino64_t st_ino;
    nlink_t st_nlink;
    mode_t st_mode;
    uid_t st_uid;
    gid_t st_gid;
    int pad0;
    dev_t st_rdev;
    off_t st_size;
    struct timespec st_atim;
    struct timespec st_mtim;
    struct timespec st_ctim;
    blksize_t st_blksize;
    blkcnt64_t st_blocks;
    long int __unused[3];
};

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 __xmknod(int, const char *, mode_t, dev_t *);
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);
extern int fchmod(int, mode_t);
extern mode_t umask(mode_t);
```

11.3.60 sys/statvfs.h

```

struct statvfs {
    unsigned long int f_bsize;
    unsigned long int f_frsize;
    fsblkcnt64_t f_blocks;
    fsblkcnt64_t f_bfree;
```

```
967             fsblkcnt64_t f_bavail;
968             fsfilcnt64_t f_files;
969             fsfilcnt64_t f_ffree;
970             fsfilcnt64_t f_favail;
971             unsigned long int f_fsid;
972             unsigned long int f_flag;
973             unsigned long int f_namemax;
974             int __f_spare[6];
975         };
976         struct statvfs64 {
977             unsigned long int f_bsize;
978             unsigned long int f_frsize;
979             fsblkcnt64_t f_blocks;
980             fsblkcnt64_t f_bfree;
981             fsblkcnt64_t f_bavail;
982             fsfilcnt64_t f_files;
983             fsfilcnt64_t f_ffree;
984             fsfilcnt64_t f_favail;
985             unsigned long int f_fsid;
986             unsigned long int f_flag;
987             unsigned long int f_namemax;
988             int __f_spare[6];
989         };
990         extern int fstatvfs(int, struct statvfs *);
991         extern int fstatvfs64(int, struct statvfs64 *);
992         extern int statvfs(const char *, struct statvfs *);
993         extern int statvfs64(const char *, struct statvfs64 *);
```

11.3.61 sys/time.h

```
994
995         extern int getitimer(__itimer_which_t, struct itimerval *);
996         extern int setitimer(__itimer_which_t, const struct itimerval *,
997                              struct itimerval *);
998         extern int adjtime(const struct timeval *, struct timeval *);
999         extern int gettimeofday(struct timeval *, struct timezone *);
1000        extern int utimes(const char *, const struct timeval *);
```

11.3.62 sys/timeb.h

```
1001
1002        extern int ftime(struct timeb *);
```

11.3.63 sys/times.h

```
1003
1004        extern clock_t times(struct tms *);
```

11.3.64 sys/types.h

```
1005
1006        typedef long int int64_t;
1007
1008        typedef int64_t ssize_t;
1009
1010        #define __FDSET_LONGS   16
```

11.3.65 sys/uio.h

```
1011
1012        extern ssize_t readv(int, const struct iovec *, int);
1013        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 *);
```

11.3.68 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 CR2     1024
#define CR3     1536
#define CRDLY   1536
#define VT1     16384
#define VTDLY   16384
#define OLCUC   2
#define TAB1    2048
#define NLDLY   256
#define FF1     32768
#define FFDLY   32768
#define ONLCR   4
#define XCASE   4
#define TAB2    4096
#define CR1     512
#define IUCLC   512
#define TAB3    6144
#define TABDLY  6144
#define BS1     8192
#define BSDLY   8192

#define VSUSP   10
#define VEOL    11
#define VREPRINT        12
#define VDISCARD        13
#define VWERASE 14
#define VEOL2   16
#define VMIN    6
#define VSWTC   7
#define VSTART  8
#define VSTOP   9
```

```
#define IXON    1024
#define IXOFF   4096

#define HUPCL   1024
#define CREAD   128
#define CS6     16
#define CLOCAL  2048
#define PARENB  256
#define CS7     32
#define CS8     48
#define CSIZE   48
#define VTIME   5
#define PARODD  512
#define CSTOPB  64

#define ISIG    1
#define ECHOPRT 1024
#define NOFLSH  128
#define ECHOE   16
#define PENDIN  16384
#define ICANON  2
#define ECHOKE  2048
#define TOSTOP  256
#define ECHOK   32
#define IEXTEN  32768
#define FLUSHO  4096
#define ECHOCTL 512
#define ECHONL  64

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 *);
```

```
extern int daylight(void);
extern long int timezone(void);
extern char *tzname(void);
extern void tzset(void);
extern char *asctime_r(const struct tm *, char *);
extern struct tm *gmtime_r(const time_t *, struct tm *);
extern struct tm *localtime_r(const time_t *, struct tm *);
extern int clock_getcpuclockid(pid_t, clockid_t *);
extern int clock_getres(clockid_t, struct timespec *);
extern int clock_gettime(clockid_t, struct timespec *);
extern int clock_nanosleep(clockid_t, int, const struct timespec *,
                           struct timespec *);
extern int clock_settime(clockid_t, const struct timespec *);
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

```

#define NGREG   27

typedef union {
    double d;
    float f;
} fpreg_t;

typedef struct {
    unsigned int fpc;
    fpreg_t fprs[16];
} fpregset_t;

typedef struct {
    _psw_t psw;
    unsigned long int gregs[16];
    unsigned int aregs[16];
    fpregset_t fpregs;
} 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;
} 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

```
```

```
1178            extern char **__environ(void);
1179            extern pid_t __getpgid(pid_t);
1180            extern void _exit(int);
1181            extern int acct(const char *);
1182            extern unsigned int alarm(unsigned int);
1183            extern int chown(const char *, uid_t, gid_t);
1184            extern int chroot(const char *);
1185            extern size_t confstr(int, char *, size_t);
1186            extern int creat(const char *, mode_t);
1187            extern int creat64(const char *, mode_t);
1188            extern char *ctermid(char *);
1189            extern char *cuserid(char *);
1190            extern int daemon(int, int);
1191            extern int execl(const char *, const char *, ...);
1192            extern int execle(const char *, const char *, ...);
1193            extern int execlp(const char *, const char *, ...);
1194            extern int execv(const char *, char *const);
1195            extern int execvp(const char *, char *const);
1196            extern int fdatasync(int);
1197            extern int ftruncate64(int, off64_t);
1198            extern long int gethostid(void);
1199            extern char *getlogin(void);
1200            extern int getlogin_r(char *, size_t);
1201            extern int getopt(int, char *const, const char *);
1202            extern pid_t getpgrp(void);
1203            extern pid_t getsid(pid_t);
1204            extern char *getwd(char *);
1205            extern int lockf(int, int, off_t);
1206            extern int mkstemp(char *);
1207            extern int nice(int);
1208            extern char *optarg(void);
1209            extern int opterr(void);
1210            extern int optind(void);
1211            extern int optopt(void);
1212            extern int rename(const char *, const char *);
1213            extern int setegid(gid_t);
1214            extern int seteuid(uid_t);
1215            extern int sethostname(const char *, size_t);
1216            extern int setpgrp(void);
1217            extern void swab(const void *, void *, ssize_t);
1218            extern void sync(void);
1219            extern pid_t tcgetpgrp(int);
1220            extern int tcsetpgrp(int, pid_t);
1221            extern int truncate(const char *, off_t);
1222            extern int truncate64(const char *, off64_t);
1223            extern char *ttyname(int);
1224            extern unsigned int ualarm(useconds_t, useconds_t);
1225            extern int usleep(useconds_t);
1226            extern int close(int);
1227            extern int fsync(int);
1228            extern off_t lseek(int, off_t, int);
1229            extern int open(const char *, int, ...);
1230            extern int pause(void);
1231            extern ssize_t read(int, void *, size_t);
1232            extern ssize_t write(int, const void *, size_t);
1233            extern char *crypt(char *, char *);
1234            extern void encrypt(char *, int);
1235            extern void setkey(const char *);
1236            extern int access(const char *, int);
1237            extern int brk(void *);
1238            extern int chdir(const char *);
1239            extern int dup(int);
1240            extern int dup2(int, int);
1241            extern int execve(const char *, char *const, char *const);
```

```
extern int fchdir(int);
extern int fchown(int, uid_t, gid_t);
extern pid_t fork(void);
extern gid_t getegid(void);
extern uid_t geteuid(void);
extern gid_t getgid(void);
extern int getgroups(int, gid_t);
extern int gethostname(char *, size_t);
extern pid_t getpgid(pid_t);
extern pid_t getpid(void);
extern uid_t getuid(void);
extern int lchown(const char *, uid_t, gid_t);
extern int link(const char *, const char *);
extern int mkdir(const char *, mode_t);
extern long int pathconf(const char *, int);
extern int pipe(int);
extern int readlink(const char *, char *, size_t);
extern int rmdir(const char *);
extern void *sbrk(ptrdiff_t);
extern int select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
extern int setgid(gid_t);
extern int setpgid(pid_t, pid_t);
extern int setregid(gid_t, gid_t);
extern int setreuid(uid_t, uid_t);
extern pid_t setsid(void);
extern int setuid(uid_t);
extern unsigned int sleep(unsigned int);
extern int symlink(const char *, const char *);
extern long int sysconf(int);
extern int unlink(const char *);
extern pid_t vfork(void);
extern ssize_t pread(int, void *, size_t, off_t);
extern ssize_t pwrite(int, const void *, size_t, off_t);
extern char **_environ(void);
extern long int fpathconf(int, int);
extern int ftruncate(int, off_t);
extern char *getcwd(char *, size_t);
extern int getpagesize(void);
extern pid_t getppid(void);
extern int isatty(int);
extern loff_t lseek64(int, loff_t, int);
extern int open64(const char *, int, ...);
extern ssize_t pread64(int, void *, size_t, off64_t);
extern ssize_t pwrite64(int, const void *, size_t, off64_t);
extern int ttyname_r(int, char *, size_t);
```

11.3.75 utime.h

```

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

11.3.76 utmp.h

```

struct lastlog {
    time_t ll_time;
    char ll_line[UT_LINESIZE];
    char ll_host[UT_HOSTSIZE];
};

struct utmp {
    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];
};

extern void endutent(void);
extern struct utmp *getutent(void);
extern void setutent(void);
extern int getutent_r(struct utmp *, struct utmp **);
extern int utmpname(const char *);
extern int login_tty(int);
extern void login(const struct utmp *);
extern int logout(const char *);
extern void logwtmp(const char *, const char *, const char *);
```

11.3.77 utmpx.h

```

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];
};

extern void endutxent(void);
extern struct utmpx *getutxent(void);
extern struct utmpx *getutxid(const struct utmpx *);
extern struct utmpx *getutxline(const struct utmpx *);
extern struct utmpx *pututxline(const struct utmpx *);
extern void setutxent(void);
```

11.3.78 wchar.h

```

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 *wcscpy(wchar_t *, const wchar_t *);
extern size_t wcscspn(const wchar_t *, const wchar_t *);
extern wchar_t *wcsdup(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);
```

```
1358            extern wchar_t *wcsncpy(wchar_t *, const wchar_t *, size_t);
1359            extern wchar_t *wcspbrk(const wchar_t *, const wchar_t *);
1360            extern wchar_t *wcsrchr(const wchar_t *, wchar_t);
1361            extern size_t wcsspn(const wchar_t *, const wchar_t *);
1362            extern wchar_t *wcsstr(const wchar_t *, const wchar_t *);
1363            extern wchar_t *wcstok(wchar_t *, const wchar_t *, wchar_t * *);
1364            extern int wcswidth(const wchar_t *, size_t);
1365            extern size_t wcsxfrm(wchar_t *, const wchar_t *, size_t);
1366            extern int wctob(wint_t);
1367            extern int wcwidth(wchar_t);
1368            extern wchar_t *wmemchr(const wchar_t *, wchar_t, size_t);
1369            extern int wmemcmp(const wchar_t *, const wchar_t *, size_t);
1370            extern wchar_t *wmemcpy(wchar_t *, const wchar_t *, size_t);
1371            extern wchar_t *wmemmove(wchar_t *, const wchar_t *, size_t);
1372            extern wchar_t *wmemset(wchar_t *, wchar_t, size_t);
1373            extern size_t mbrlen(const char *, size_t, mbstate_t *);
1374            extern size_t mbrtowc(wchar_t *, const char *, size_t, mbstate_t *);
1375            extern int mbsinit(const mbstate_t *);
1376            extern size_t mbsnrtowcs(wchar_t *, const char **, size_t, size_t,
1377                                     mbstate_t *);
1378            extern size_t mbsrtowcs(wchar_t *, const char **, size_t, mbstate_t *);
1379            extern wchar_t *wcpcpy(wchar_t *, const wchar_t *);
1380            extern wchar_t *wcpncpy(wchar_t *, const wchar_t *, size_t);
1381            extern size_t wcrtomb(char *, wchar_t, mbstate_t *);
1382            extern size_t wcslen(const wchar_t *);
1383            extern size_t wcsnrtombs(char *, const wchar_t * *, size_t, size_t,
1384                                     mbstate_t *);
1385            extern size_t wcsrtombs(char *, const wchar_t * *, size_t, mbstate_t *);
1386            extern double wcstod(const wchar_t *, wchar_t * *);
1387            extern float wcstof(const wchar_t *, wchar_t * *);
1388            extern long int wcstol(const wchar_t *, wchar_t * *, int);
1389            extern long double wcstold(const wchar_t *, wchar_t * *);
1390            extern long long int wcstoq(const wchar_t *, wchar_t * *, int);
1391            extern unsigned long int wcstoul(const wchar_t *, wchar_t * *, int);
1392            extern unsigned long long int wcstouq(const wchar_t *, wchar_t * *, int);
1393            extern wchar_t *wcswcs(const wchar_t *, const wchar_t *);
1394            extern int wcscasecmp(const wchar_t *, const wchar_t *);
1395            extern int wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
1396            extern size_t wcsnlen(const wchar_t *, size_t);
1397            extern long long int wcstoll(const wchar_t *, wchar_t * *, int);
1398            extern unsigned long long int wcstoull(const wchar_t *, wchar_t * *, int);
1399            extern wint_t btowc(int);
1400            extern wint_t fgetwc(FILE *);
1401            extern wint_t fgetwc_unlocked(FILE *);
1402            extern wchar_t *fgetws(wchar_t *, int, FILE *);
1403            extern wint_t fputwc(wchar_t, FILE *);
1404            extern int fputws(const wchar_t *, FILE *);
1405            extern int fwide(FILE *, int);
1406            extern int fwprintf(FILE *, const wchar_t *, ...);
1407            extern int fwscanf(FILE *, const wchar_t *, ...);
1408            extern wint_t getwc(FILE *);
1409            extern wint_t getwchar(void);
1410            extern wint_t putwc(wchar_t, FILE *);
1411            extern wint_t putwchar(wchar_t);
1412            extern int swprintf(wchar_t *, size_t, const wchar_t *, ...);
1413            extern int swscanf(const wchar_t *, const wchar_t *, ...);
1414            extern wint_t ungetwc(wint_t, FILE *);
1415            extern int vfwprintf(FILE *, const wchar_t *, va_list);
1416            extern int vfwscanf(FILE *, const wchar_t *, va_list);
1417            extern int vswprintf(wchar_t *, size_t, const wchar_t *, va_list);
1418            extern int vswscanf(const wchar_t *, const wchar_t *, va_list);
1419            extern int vwprintf(const wchar_t *, va_list);
1420            extern int vwscanf(const wchar_t *, va_list);
1421            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

Library:	libm
SONAME:	libm.so.6

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

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

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

__finite(GLIBC_2.	__finitef(GLIBC_2	__finitel(GLIBC_2	__fpclassify(GLIB

2) [ISOC99]	.2) [ISOC99]	.2) [ISOC99]	C_2.2) [LSB]
__fpclassifyf(GLIBC_2.2) [LSB]	acos(GLIBC_2.2) [SUSv3]	acosf(GLIBC_2.2) [SUSv3]	acosh(GLIBC_2.2) [SUSv3]
acoshf(GLIBC_2.2) [SUSv3]	acoshl(GLIBC_2.2) [SUSv3]	acosl(GLIBC_2.2) [SUSv3]	asin(GLIBC_2.2) [SUSv3]
asinf(GLIBC_2.2) [SUSv3]	asinh(GLIBC_2.2) [SUSv3]	asinhf(GLIBC_2.2) [SUSv3]	asinhl(GLIBC_2.2) [SUSv3]
asinl(GLIBC_2.2) [SUSv3]	atan(GLIBC_2.2) [SUSv3]	atan2(GLIBC_2.2) [SUSv3]	atan2f(GLIBC_2.2) [SUSv3]
atan2l(GLIBC_2.2) [SUSv3]	atanf(GLIBC_2.2) [SUSv3]	atanh(GLIBC_2.2) [SUSv3]	atanhf(GLIBC_2.2) [SUSv3]
atanhl(GLIBC_2.2) [SUSv3]	atanl(GLIBC_2.2) [SUSv3]	cabs(GLIBC_2.2) [SUSv3]	cabsf(GLIBC_2.2) [SUSv3]
cabsl(GLIBC_2.2) [SUSv3]	cacos(GLIBC_2.2) [SUSv3]	cacosf(GLIBC_2.2) [SUSv3]	cacosh(GLIBC_2.2) [SUSv3]
cacoshf(GLIBC_2.2) [SUSv3]	cacoshl(GLIBC_2.2) [SUSv3]	cacosl(GLIBC_2.2) [SUSv3]	carg(GLIBC_2.2) [SUSv3]
cargf(GLIBC_2.2) [SUSv3]	cargl(GLIBC_2.2) [SUSv3]	casin(GLIBC_2.2) [SUSv3]	casinf(GLIBC_2.2) [SUSv3]
casinh(GLIBC_2.2) [SUSv3]	casinhf(GLIBC_2.2) [SUSv3]	casinhl(GLIBC_2.2) [SUSv3]	casinl(GLIBC_2.2) [SUSv3]
catan(GLIBC_2.2) [SUSv3]	catanf(GLIBC_2.2) [SUSv3]	catanh(GLIBC_2.2) [SUSv3]	catanhf(GLIBC_2.2) [SUSv3]
catanhl(GLIBC_2.2) [SUSv3]	catanl(GLIBC_2.2) [SUSv3]	cbrt(GLIBC_2.2) [SUSv3]	cbrtf(GLIBC_2.2) [SUSv3]
cbrtl(GLIBC_2.2) [SUSv3]	ccos(GLIBC_2.2) [SUSv3]	ccosf(GLIBC_2.2) [SUSv3]	ccosh(GLIBC_2.2) [SUSv3]
ccoshf(GLIBC_2.2) [SUSv3]	ccoshl(GLIBC_2.2) [SUSv3]	ccosl(GLIBC_2.2) [SUSv3]	ceil(GLIBC_2.2) [SUSv3]
ceilf(GLIBC_2.2) [SUSv3]	ceill(GLIBC_2.2) [SUSv3]	cexp(GLIBC_2.2) [SUSv3]	cexpf(GLIBC_2.2) [SUSv3]
cexpl(GLIBC_2.2) [SUSv3]	cimag(GLIBC_2.2) [SUSv3]	cimagf(GLIBC_2.2) [SUSv3]	cimagl(GLIBC_2.2) [SUSv3]
clog(GLIBC_2.2) [SUSv3]	clog10(GLIBC_2.2) [ISOC99]	clog10f(GLIBC_2.2) [ISOC99]	clog10l(GLIBC_2.2) [ISOC99]
clogf(GLIBC_2.2) [SUSv3]	clogl(GLIBC_2.2) [SUSv3]	conj(GLIBC_2.2) [SUSv3]	conjf(GLIBC_2.2) [SUSv3]
conjl(GLIBC_2.2) [SUSv3]	copysign(GLIBC_2.2) [SUSv3]	copysignf(GLIBC_2.2) [SUSv3]	copysignl(GLIBC_2.2) [SUSv3]
cos(GLIBC_2.2) [SUSv3]	cosf(GLIBC_2.2) [SUSv3]	cosh(GLIBC_2.2) [SUSv3]	coshf(GLIBC_2.2) [SUSv3]

coshl(GLIBC_2.2) [SUSv3]	cosl(GLIBC_2.2) [SUSv3]	cpow(GLIBC_2.2) [SUSv3]	cpowf(GLIBC_2.2) [SUSv3]
cpowl(GLIBC_2.2) [SUSv3]	cproj(GLIBC_2.2) [SUSv3]	cprojf(GLIBC_2.2) [SUSv3]	cprojl(GLIBC_2.2) [SUSv3]
creal(GLIBC_2.2) [SUSv3]	crealf(GLIBC_2.2) [SUSv3]	creall(GLIBC_2.2) [SUSv3]	csin(GLIBC_2.2) [SUSv3]
csinf(GLIBC_2.2) [SUSv3]	csinh(GLIBC_2.2) [SUSv3]	csinhf(GLIBC_2.2) [SUSv3]	csinhl(GLIBC_2.2) [SUSv3]
csinl(GLIBC_2.2) [SUSv3]	csqrt(GLIBC_2.2) [SUSv3]	csqrtf(GLIBC_2.2) [SUSv3]	csqrtl(GLIBC_2.2) [SUSv3]
ctan(GLIBC_2.2) [SUSv3]	ctanf(GLIBC_2.2) [SUSv3]	ctanh(GLIBC_2.2) [SUSv3]	ctanhf(GLIBC_2.2) [SUSv3]
ctanhl(GLIBC_2.2) [SUSv3]	ctanl(GLIBC_2.2) [SUSv3]	dremf(GLIBC_2.2) [ISOC99]	dreml(GLIBC_2.2) [ISOC99]
erf(GLIBC_2.2) [SUSv3]	erfc(GLIBC_2.2) [SUSv3]	erfcf(GLIBC_2.2) [SUSv3]	erfcl(GLIBC_2.2) [SUSv3]
erff(GLIBC_2.2) [SUSv3]	erfl(GLIBC_2.2) [SUSv3]	exp(GLIBC_2.2) [SUSv3]	exp2(GLIBC_2.2) [SUSv3]
exp2f(GLIBC_2.2) [SUSv3]	expf(GLIBC_2.2) [SUSv3]	expl(GLIBC_2.2) [SUSv3]	expm1(GLIBC_2.2) [SUSv3]
expm1f(GLIBC_2.2) [SUSv3]	expm1l(GLIBC_2.2) [SUSv3]	fabs(GLIBC_2.2) [SUSv3]	fabsf(GLIBC_2.2) [SUSv3]
fabsl(GLIBC_2.2) [SUSv3]	fdim(GLIBC_2.2) [SUSv3]	fdimf(GLIBC_2.2) [SUSv3]	fdiml(GLIBC_2.2) [SUSv3]
feclearexcept(GLIBC_2.2) [SUSv3]	fegetenv(GLIBC_2.2) [SUSv3]	fegetexceptflag(GLIBC_2.2) [SUSv3]	fegetround(GLIBC_2.2) [SUSv3]
feholdexcept(GLIBC_2.2) [SUSv3]	feraiseexcept(GLIBC_2.2) [SUSv3]	fesetenv(GLIBC_2.2) [SUSv3]	fesetexceptflag(GLIBC_2.2) [SUSv3]
fesetround(GLIBC_2.2) [SUSv3]	fetestexcept(GLIBC_2.2) [SUSv3]	feupdateenv(GLIBC_2.2) [SUSv3]	finite(GLIBC_2.2) [SUSv2]
finitef(GLIBC_2.2) [ISOC99]	finitel(GLIBC_2.2) [ISOC99]	floor(GLIBC_2.2) [SUSv3]	floorf(GLIBC_2.2) [SUSv3]
floorl(GLIBC_2.2) [SUSv3]	fma(GLIBC_2.2) [SUSv3]	fmaf(GLIBC_2.2) [SUSv3]	fmal(GLIBC_2.2) [SUSv3]
fmax(GLIBC_2.2) [SUSv3]	fmaxf(GLIBC_2.2) [SUSv3]	fmaxl(GLIBC_2.2) [SUSv3]	fmin(GLIBC_2.2) [SUSv3]
fminf(GLIBC_2.2) [SUSv3]	fminl(GLIBC_2.2) [SUSv3]	fmod(GLIBC_2.2) [SUSv3]	fmodf(GLIBC_2.2) [SUSv3]
fmodl(GLIBC_2.2) [SUSv3]	frexp(GLIBC_2.2) [SUSv3]	frexpf(GLIBC_2.2) [SUSv3]	frexpl(GLIBC_2.2) [SUSv3]
gamma(GLIBC_2.	gammaf(GLIBC_2	gammal(GLIBC_2	hypot(GLIBC_2.2)

2) [SUSv2]	.2) [ISOC99]	.2) [ISOC99]	[SUSv3]
hypotf(GLIBC_2.2) [SUSv3]	hypotl(GLIBC_2.2) [SUSv3]	ilogb(GLIBC_2.2) [SUSv3]	ilogbf(GLIBC_2.2) [SUSv3]
ilogbl(GLIBC_2.2) [SUSv3]	j0(GLIBC_2.2) [SUSv3]	j0f(GLIBC_2.2) [ISOC99]	j0l(GLIBC_2.2) [ISOC99]
j1(GLIBC_2.2) [SUSv3]	j1f(GLIBC_2.2) [ISOC99]	j1l(GLIBC_2.2) [ISOC99]	jn(GLIBC_2.2) [SUSv3]
jnf(GLIBC_2.2) [ISOC99]	jnl(GLIBC_2.2) [ISOC99]	ldexp(GLIBC_2.2) [SUSv3]	ldexpf(GLIBC_2.2) [SUSv3]
ldexpl(GLIBC_2.2) [SUSv3]	lgamma(GLIBC_2.2) [SUSv3]	lgamma_r(GLIBC_2.2) [ISOC99]	lgammaf(GLIBC_2.2) [SUSv3]
lgammaf_r(GLIBC_2.2) [ISOC99]	lgammal(GLIBC_2.2) [SUSv3]	lgammal_r(GLIBC_2.2) [ISOC99]	llrint(GLIBC_2.2) [SUSv3]
llrintf(GLIBC_2.2) [SUSv3]	llrintl(GLIBC_2.2) [SUSv3]	llround(GLIBC_2.2) [SUSv3]	llroundf(GLIBC_2.2) [SUSv3]
llroundl(GLIBC_2.2) [SUSv3]	log(GLIBC_2.2) [SUSv3]	log10(GLIBC_2.2) [SUSv3]	log10f(GLIBC_2.2) [SUSv3]
log10l(GLIBC_2.2) [SUSv3]	log1p(GLIBC_2.2) [SUSv3]	log1pf(GLIBC_2.2) [SUSv3]	log1pl(GLIBC_2.2) [SUSv3]
log2(GLIBC_2.2) [SUSv3]	log2f(GLIBC_2.2) [SUSv3]	log2l(GLIBC_2.2) [SUSv3]	logb(GLIBC_2.2) [SUSv3]
logbf(GLIBC_2.2) [SUSv3]	logbl(GLIBC_2.2) [SUSv3]	logf(GLIBC_2.2) [SUSv3]	logl(GLIBC_2.2) [SUSv3]
lrint(GLIBC_2.2) [SUSv3]	lrintf(GLIBC_2.2) [SUSv3]	lrintl(GLIBC_2.2) [SUSv3]	lround(GLIBC_2.2) [SUSv3]
lroundf(GLIBC_2.2) [SUSv3]	lroundl(GLIBC_2.2) [SUSv3]	matherr(GLIBC_2.2) [ISOC99]	modf(GLIBC_2.2) [SUSv3]
modff(GLIBC_2.2) [SUSv3]	modfl(GLIBC_2.2) [SUSv3]	nan(GLIBC_2.2) [SUSv3]	nanf(GLIBC_2.2) [SUSv3]
nanl(GLIBC_2.2) [SUSv3]	nearbyint(GLIBC_2.2) [SUSv3]	nearbyintf(GLIBC_2.2) [SUSv3]	nearbyintl(GLIBC_2.2) [SUSv3]
nextafter(GLIBC_2.2) [SUSv3]	nextafterf(GLIBC_2.2) [SUSv3]	nextafterl(GLIBC_2.2) [SUSv3]	nexttoward(GLIBC_2.2) [SUSv3]
nexttowardf(GLIBC_2.2) [SUSv3]	nexttowardl(GLIBC_2.2) [SUSv3]	pow(GLIBC_2.2) [SUSv3]	pow10(GLIBC_2.2) [ISOC99]
pow10f(GLIBC_2.2) [ISOC99]	pow10l(GLIBC_2.2) [ISOC99]	powf(GLIBC_2.2) [SUSv3]	powl(GLIBC_2.2) [SUSv3]
remainder(GLIBC_2.2) [SUSv3]	remainderf(GLIBC_2.2) [SUSv3]	remainderl(GLIBC_2.2) [SUSv3]	remquo(GLIBC_2.2) [SUSv3]
remquof(GLIBC_2.2) [SUSv3]	remquol(GLIBC_2.2) [SUSv3]	rint(GLIBC_2.2) [SUSv3]	rintf(GLIBC_2.2) [SUSv3]

rintl(GLIBC_2.2) [SUSv3]	round(GLIBC_2.2) [SUSv3]	roundf(GLIBC_2.2) [SUSv3]	roundl(GLIBC_2.2) [SUSv3]
scalb(GLIBC_2.2) [SUSv3]	scalbf(GLIBC_2.2) [ISOC99]	scalbl(GLIBC_2.2) [ISOC99]	scalbln(GLIBC_2.2) [SUSv3]
scalblnf(GLIBC_2. 2) [SUSv3]	scalblnl(GLIBC_2. 2) [SUSv3]	scalbn(GLIBC_2.2) [SUSv3]	scalbnf(GLIBC_2. 2) [SUSv3]
scalbnl(GLIBC_2.2) [SUSv3]	significand(GLIB C_2.2) [ISOC99]	significandf(GLIB C_2.2) [ISOC99]	significandl(GLIB C_2.2) [ISOC99]
sin(GLIBC_2.2) [SUSv3]	sincos(GLIBC_2.2) [ISOC99]	sincosf(GLIBC_2.2) [ISOC99]	sincosl(GLIBC_2.2) [ISOC99]
sinf(GLIBC_2.2) [SUSv3]	sinh(GLIBC_2.2) [SUSv3]	sinhf(GLIBC_2.2) [SUSv3]	sinhl(GLIBC_2.2) [SUSv3]
sinl(GLIBC_2.2) [SUSv3]	sqrt(GLIBC_2.2) [SUSv3]	sqrtf(GLIBC_2.2) [SUSv3]	sqrtl(GLIBC_2.2) [SUSv3]
tan(GLIBC_2.2) [SUSv3]	tanf(GLIBC_2.2) [SUSv3]	tanh(GLIBC_2.2) [SUSv3]	tanhf(GLIBC_2.2) [SUSv3]
tanhl(GLIBC_2.2) [SUSv3]	tanl(GLIBC_2.2) [SUSv3]	tgamma(GLIBC_2 .2) [SUSv3]	tgammaf(GLIBC_ 2.2) [SUSv3]
tgammal(GLIBC_ 2.2) [SUSv3]	trunc(GLIBC_2.2) [SUSv3]	truncf(GLIBC_2.2) [SUSv3]	truncl(GLIBC_2.2) [SUSv3]
y0(GLIBC_2.2) [SUSv3]	y0f(GLIBC_2.2) [ISOC99]	y0l(GLIBC_2.2) [ISOC99]	y1(GLIBC_2.2) [SUSv3]
y1f(GLIBC_2.2) [ISOC99]	y1l(GLIBC_2.2) [ISOC99]	yn(GLIBC_2.2) [SUSv3]	ynf(GLIBC_2.2) [ISOC99]
ynl(GLIBC_2.2) [ISOC99]			

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.

Table 11-26 libm - Math Data Interfaces

signgam(GLIBC_2 .2) [SUSv3]			

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

```
extern double cabs(double complex);
extern float cabsf(float complex);
extern long double cabsl(long double complex);
extern double complex cacos(double complex);
extern float complex cacosf(float complex);
extern double complex cacosh(double complex);
extern float complex cacoshf(float complex);
extern long double complex cacoshl(long double complex);
extern long double complex cacosl(long double complex);
extern double carg(double complex);
extern float cargf(float complex);
extern long double cargl(long double complex);
extern double complex casin(double complex);
extern float complex casinf(float complex);
extern double complex casinh(double complex);
extern float complex casinhf(float complex);
extern long double complex casinhl(long double complex);
extern long double complex casinl(long double complex);
extern double complex catan(double complex);
extern float complex catanf(float complex);
extern double complex catanh(double complex);
extern float complex catanhf(float complex);
extern long double complex catanhl(long double complex);
extern long double complex catanl(long double complex);
extern double complex ccos(double complex);
extern float complex ccosf(float complex);
extern double complex ccosh(double complex);
extern float complex ccoshf(float complex);
extern long double complex ccoshl(long double complex);
extern long double complex ccosl(long double complex);
extern double complex cexp(double complex);
extern float complex cexpf(float complex);
extern long double complex cexpl(long double complex);
extern double cimag(double complex);
extern float cimagf(float complex);
extern long double cimagl(long double complex);
extern double complex clog(double complex);
extern float complex clog10f(float complex);
extern long double complex clog10l(long double complex);
extern float complex clogf(float complex);
extern long double complex clogl(long double complex);
extern double complex conj(double complex);
extern float complex conjf(float complex);
extern long double complex conjl(long double complex);
extern double complex cpow(double complex, double complex);
extern float complex cpowf(float complex, 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 double creal(double complex);
```

```
extern float crealf(float complex);
extern long double creall(long double complex);
extern double complex csin(double complex);
extern float complex csinf(float complex);
extern double complex csinh(double complex);
extern float complex csinhf(float complex);
extern long double complex csinhl(long double complex);
extern long double complex csinl(long double complex);
extern double complex csqrt(double complex);
extern float complex csqrtf(float complex);
extern long double complex csqrtl(long double complex);
extern double complex ctan(double complex);
extern float complex ctanf(float complex);
extern double complex ctanh(double complex);
extern float complex ctanhf(float complex);
extern long double complex ctanhl(long double complex);
extern long double complex ctanl(long double complex);
```

11.5.2 fenv.h

```

#define FE_INEXACT      0x08
#define FE_UNDERFLOW    0x10
#define FE_OVERFLOW     0x20
#define FE_DIVBYZERO    0x40
#define FE_INVALID      0x80

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

#define FE_TONEAREST    0
#define FE_TOWARDZERO   0x1
#define FE_UPWARD       0x2
#define FE_DOWNWARD     0x3

typedef unsigned int fexcept_t;

typedef struct {
    fexcept_t fpc;
    void *ieee_instruction_pointer;
} fenv_t;

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

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

11.5.3 math.h

```

#define fpclassify(x)   \
        (sizeof (x) == sizeof (float) ? __fpclassifyf (x) : __fpclassify
(x) )
#define signbit(x)      \
```

```
                (sizeof (x) == sizeof (float)? __signbitf (x): __signbit (x))

#define FP_ILOGB0       -2147483647
#define FP_ILOGBNAN     2147483647

extern int __finite(double);
extern int __finitef(float);
extern int __finitel(long double);
extern int __isinf(double);
extern int __isinff(float);
extern int __isinfl(long double);
extern int __isnan(double);
extern int __isnanf(float);
extern int __isnanl(long double);
extern int __signbit(double);
extern int __signbitf(float);
extern int __fpclassify(double);
extern int __fpclassifyf(float);
extern int __fpclassifyl(long double);
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 acosh(double);
extern double asinh(double);
extern double atanh(double);
extern double asin(double);
extern double atan(double);
extern double atan2(double, double);
extern double cbrt(double);
extern double ceil(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);
```

```
1653            extern double y0(double);
1654            extern double y1(double);
1655            extern double yn(int, double);
1656            extern float copysignf(float, float);
1657            extern long double copysignl(long double, long double);
1658            extern int finitef(float);
1659            extern int finitel(long double);
1660            extern float frexpf(float, int *);
1661            extern long double frexpl(long double, int *);
1662            extern float ldexpf(float, int);
1663            extern long double ldexpl(long double, int);
1664            extern float modff(float, float *);
1665            extern long double modfl(long double, long double *);
1666            extern double scalbln(double, long int);
1667            extern float scalblnf(float, long int);
1668            extern long double scalblnl(long double, long int);
1669            extern double scalbn(double, int);
1670            extern float scalbnf(float, int);
1671            extern long double scalbnl(long double, int);
1672            extern float acosf(float);
1673            extern float acoshf(float);
1674            extern long double acoshl(long double);
1675            extern long double acosl(long double);
1676            extern float asinf(float);
1677            extern float asinhf(float);
1678            extern long double asinhl(long double);
1679            extern long double asinl(long double);
1680            extern float atan2f(float, float);
1681            extern long double atan2l(long double, long double);
1682            extern float atanf(float);
1683            extern float atanhf(float);
1684            extern long double atanhl(long double);
1685            extern long double atanl(long double);
1686            extern float cbrtf(float);
1687            extern long double cbrtl(long double);
1688            extern float ceilf(float);
1689            extern long double ceill(long double);
1690            extern float cosf(float);
1691            extern float coshf(float);
1692            extern long double coshl(long double);
1693            extern long double cosl(long double);
1694            extern float dremf(float, float);
1695            extern long double dreml(long double, long double);
1696            extern float erfcf(float);
1697            extern long double erfcl(long double);
1698            extern float erff(float);
1699            extern long double erfl(long double);
1700            extern double exp2(double);
1701            extern float exp2f(float);
1702            extern long double exp2l(long double);
1703            extern float expf(float);
1704            extern long double expl(long double);
1705            extern float expm1f(float);
1706            extern long double expm1l(long double);
1707            extern float fabsf(float);
1708            extern long double fabsl(long double);
1709            extern double fdim(double, double);
1710            extern float fdimf(float, float);
1711            extern long double fdiml(long double, long double);
1712            extern float floorf(float);
1713            extern long double floorl(long double);
1714            extern double fma(double, double, double);
1715            extern float fmaf(float, float, float);
1716            extern long double fmal(long double, long double, long double);
```

```
1717            extern double fmax(double, double);
1718            extern float fmaxf(float, float);
1719            extern long double fmaxl(long double, long double);
1720            extern double fmin(double, double);
1721            extern float fminf(float, float);
1722            extern long double fminl(long double, long double);
1723            extern float fmodf(float, float);
1724            extern long double fmodl(long double, long double);
1725            extern float gammaf(float);
1726            extern long double gammal(long double);
1727            extern float hypotf(float, float);
1728            extern long double hypotl(long double, long double);
1729            extern int ilogbf(float);
1730            extern int ilogbl(long double);
1731            extern float j0f(float);
1732            extern long double j0l(long double);
1733            extern float j1f(float);
1734            extern long double j1l(long double);
1735            extern float jnf(int, float);
1736            extern long double jnl(int, long double);
1737            extern double lgamma_r(double, int *);
1738            extern float lgammaf(float);
1739            extern float lgammaf_r(float, int *);
1740            extern long double lgammal(long double);
1741            extern long double lgammal_r(long double, int *);
1742            extern long long int llrint(double);
1743            extern long long int llrintf(float);
1744            extern long long int llrintl(long double);
1745            extern long long int llround(double);
1746            extern long long int llroundf(float);
1747            extern long long int llroundl(long double);
1748            extern float log10f(float);
1749            extern long double log10l(long double);
1750            extern float log1pf(float);
1751            extern long double log1pl(long double);
1752            extern double log2(double);
1753            extern float log2f(float);
1754            extern long double log2l(long double);
1755            extern float logbf(float);
1756            extern long double logbl(long double);
1757            extern float logf(float);
1758            extern long double logl(long double);
1759            extern long int lrint(double);
1760            extern long int lrintf(float);
1761            extern long int lrintl(long double);
1762            extern long int lround(double);
1763            extern long int lroundf(float);
1764            extern long int lroundl(long double);
1765            extern int matherr(struct exception *);
1766            extern double nan(const char *);
1767            extern float nanf(const char *);
1768            extern long double nanl(const char *);
1769            extern double nearbyint(double);
1770            extern float nearbyintf(float);
1771            extern long double nearbyintl(long double);
1772            extern float nextafterf(float, float);
1773            extern long double nextafterl(long double, long double);
1774            extern double nexttoward(double, long double);
1775            extern float nexttowardf(float, long double);
1776            extern long double nexttowardl(long double, long double);
1777            extern double pow10(double);
1778            extern float pow10f(float);
1779            extern long double pow10l(long double);
1780            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 float rintf(float);
extern long double rintl(long double);
extern double round(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 sinl(long double);
extern float sqrtf(float);
extern long double sqrtl(long double);
extern float tanf(float);
extern float tanhf(float);
extern long double tanhl(long double);
extern long double tanl(long double);
extern double tgamma(double);
extern float tgammaf(float);
extern long double tgammal(long double);
extern double trunc(double);
extern float truncf(float);
extern long double truncl(long double);
extern float y0f(float);
extern long double y0l(long double);
extern float y1f(float);
extern long double y1l(long double);
extern float ynf(int, float);
extern long double ynl(int, long double);
extern int __fpclassifyl(long double);
extern int __fpclassifyl(long double);
extern int __signbitl(long double);
extern int __signbitl(long double);
extern int __signbitl(long double);
extern long double exp2l(long double);
extern long double exp2l(long double);
```

11.6 Interfaces for libpthread

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

Table 11-27 libpthread Definition

Library:	libpthread
SONAME:	libpthread.so.0

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

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

11.6.1 Realtime Threads

11.6.1.1 Interfaces for Realtime Threads

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

Table 11-28 libpthread - Realtime Threads Function Interfaces

pthread_attr_getinheritsched(GLIBC_2.2) [SUSv3]	pthread_attr_getschedpolicy(GLIBC_2.2) [SUSv3]	pthread_attr_getscope(GLIBC_2.2) [SUSv3]	pthread_attr_setinheritsched(GLIBC_2.2) [SUSv3]
pthread_attr_setschedpolicy(GLIBC_2.2) [SUSv3]	pthread_attr_setscope(GLIBC_2.2) [SUSv3]	pthread_getschedparam(GLIBC_2.2) [SUSv3]	pthread_setschedparam(GLIBC_2.2) [SUSv3]

11.6.2 Advanced Realtime Threads

11.6.2.1 Interfaces for Advanced Realtime Threads

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

11.6.3 Posix Threads

11.6.3.1 Interfaces for Posix Threads

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

Table 11-29 libpthread - Posix Threads Function Interfaces

_pthread_cleanup_pop(GLIBC_2.2) [LSB]	_pthread_cleanup_push(GLIBC_2.2) [LSB]	pthread_attr_destroy(GLIBC_2.2) [SUSv3]	pthread_attr_getdetachstate(GLIBC_2.2) [SUSv3]
pthread_attr_getguardsize(GLIBC_2.2) [SUSv3]	pthread_attr_getschedparam(GLIBC_2.2) [SUSv3]	pthread_attr_getstack(GLIBC_2.2) [SUSv3]	pthread_attr_getstackaddr(GLIBC_2.2) [SUSv3]
pthread_attr_getstacksize(GLIBC_2.2) [SUSv3]	pthread_attr_init(GLIBC_2.2) [SUSv3]	pthread_attr_setdetachstate(GLIBC_2.2) [SUSv3]	pthread_attr_setguardsize(GLIBC_2.2) [SUSv3]
pthread_attr_setschedparam(GLIBC_2.2) [SUSv3]	pthread_attr_setstackaddr(GLIBC_2.2) [SUSv3]	pthread_attr_setstacksize(GLIBC_2.2) [SUSv3]	pthread_cancel(GLIBC_2.2) [SUSv3]
pthread_cond_broadcast(GLIBC_2.3.2) [SUSv3]	pthread_cond_destroy(GLIBC_2.3.2) [SUSv3]	pthread_cond_init(GLIBC_2.3.2) [SUSv3]	pthread_cond_signal(GLIBC_2.3.2) [SUSv3]

pthread_cond_timedwait(GLIBC_2.3.2) [SUSv3]	pthread_cond_wait(GLIBC_2.3.2) [SUSv3]	pthread_condattr_destroy(GLIBC_2.2) [SUSv3]	pthread_condattr_getpshared(GLIBC_2.2) [SUSv3]
pthread_condattr_init(GLIBC_2.2) [SUSv3]	pthread_condattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_create(GLIBC_2.2) [SUSv3]	pthread_detach(GLIBC_2.2) [SUSv3]
pthread_equal(GLIBC_2.2) [SUSv3]	pthread_exit(GLIBC_2.2) [SUSv3]	pthread_getconcurrency(GLIBC_2.2) [SUSv3]	pthread_getspecific(GLIBC_2.2) [SUSv3]
pthread_join(GLIBC_2.2) [SUSv3]	pthread_key_create(GLIBC_2.2) [SUSv3]	pthread_key_delete(GLIBC_2.2) [SUSv3]	pthread_kill(GLIBC_2.2) [SUSv3]
pthread_mutex_destroy(GLIBC_2.2) [SUSv3]	pthread_mutex_init(GLIBC_2.2) [SUSv3]	pthread_mutex_lock(GLIBC_2.2) [SUSv3]	pthread_mutex_trylock(GLIBC_2.2) [SUSv3]
pthread_mutex_unlock(GLIBC_2.2) [SUSv3]	pthread_mutexattr_destroy(GLIBC_2.2) [SUSv3]	pthread_mutexattr_getpshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_gettype(GLIBC_2.2) [SUSv3]
pthread_mutexattr_init(GLIBC_2.2) [SUSv3]	pthread_mutexattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_mutexattr_settype(GLIBC_2.2) [SUSv3]	pthread_once(GLIBC_2.2) [SUSv3]
pthread_rwlock_destroy(GLIBC_2.2) [SUSv3]	pthread_rwlock_init(GLIBC_2.2) [SUSv3]	pthread_rwlock_rdlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_timedrdlock(GLIBC_2.2) [SUSv3]
pthread_rwlock_timedwrlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_tryrdlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_trywrlock(GLIBC_2.2) [SUSv3]	pthread_rwlock_unlock(GLIBC_2.2) [SUSv3]
pthread_rwlock_wrlock(GLIBC_2.2) [SUSv3]	pthread_rwlockattr_destroy(GLIBC_2.2) [SUSv3]	pthread_rwlockattr_getpshared(GLIBC_2.2) [SUSv3]	pthread_rwlockattr_init(GLIBC_2.2) [SUSv3]
pthread_rwlockattr_setpshared(GLIBC_2.2) [SUSv3]	pthread_self(GLIBC_2.2) [SUSv3]	pthread_setcancelstate(GLIBC_2.2) [SUSv3]	pthread_setcanceltype(GLIBC_2.2) [SUSv3]
pthread_setconcurrency(GLIBC_2.2) [SUSv3]	pthread_setspecific(GLIBC_2.2) [SUSv3]	pthread_sigmask(GLIBC_2.2) [SUSv3]	pthread_testcancel(GLIBC_2.2) [SUSv3]
sem_close(GLIBC_2.2) [SUSv3]	sem_destroy(GLIBC_2.2) [SUSv3]	sem_getvalue(GLIBC_2.2) [SUSv3]	sem_init(GLIBC_2.2) [SUSv3]
sem_open(GLIBC_2.2) [SUSv3]	sem_post(GLIBC_2.2) [SUSv3]	sem_timedwait(GLIBC_2.2) [SUSv3]	sem_trywait(GLIBC_2.2) [SUSv3]
sem_unlink(GLIBC_2.2) [SUSv3]	sem_wait(GLIBC_2.2) [SUSv3]		

11.6.4 Thread aware versions of libc interfaces

11.6.4.1 Interfaces for Thread aware versions of libc interfaces

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

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

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

11.7 Data Definitions for libpthread

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

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

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

11.7.1 pthread.h

```
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 *, 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_setscope(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;
                                 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 *);
        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 int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
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;
                                        int __inheritsched;
                                        int __scope;
                                        size_t __guardsize;
                                        int __stackaddr_set;
                                        void *__stackaddr;
                                        unsigned long int __stacksize;}
                                        pthread_attr_t *, size_t *);
extern int pthread_attr_setguardsize(pthread_attr_t *,
                                        typedef unsigned long int
size_t);
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;}
                                        pthread_attr_t *, void **);
extern int pthread_attr_setstacksize(pthread_attr_t *,
                                        typedef 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;}
                                        pthread_attr_t *, size_t *);
extern int pthread_mutexattr_gettype(const typedef struct {
                                        int __mutexkind;}
                                        pthread_mutexattr_t *, int *);
extern int pthread_mutexattr_settype(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;}
                                        pthread_attr_t *, void **, size_t *);
        extern int pthread_attr_setstack(pthread_attr_t *, void *,
                                        typedef unsigned long int size_t);
        extern int pthread_condattr_getpshared(const typedef struct {
                                          int __dummy;}
                                          pthread_condattr_t *, int *);
        extern int pthread_condattr_setpshared(pthread_condattr_t *, int);
        extern int pthread_mutexattr_getpshared(const typedef struct {
                                           int __mutexkind;}
                                           pthread_mutexattr_t *, int *);
        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);
```

11.7.2 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 *);
```

11.8 Interfaces for libgcc_s

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

Table 11-31 libgcc_s Definition

Library:	libgcc_s
SONAME:	libgcc_s.so.1

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

[LSB] This Specification

11.8.1 Unwind Library

11.8.1.1 Interfaces for Unwind Library

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

Table 11-32 libgcc_s - Unwind Library Function Interfaces

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

11.9 Data Definitions for libgcc_s

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

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

This specification uses the 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.

11.9.1 unwind.h

```
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);
```

```
2166            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2167            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2168            _Unwind_Context
2169                                                               *);
2170            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2171            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2172            _Unwind_Exception
2173                                                              *);
2174            extern void _Unwind_Resume(struct _Unwind_Exception *);
2175            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2176            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2177            extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2178            extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2179            extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2180                                                    _Unwind_Stop_Fn, void *);
2181            extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2182            extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2183            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2184            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2185            _Unwind_Context
2186                                                               *);
2187            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2188            extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2189            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2190            _Unwind_Exception
2191                                                              *);
2192            extern void _Unwind_Resume(struct _Unwind_Exception *);
2193            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2194            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2195            extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2196                                                    _Unwind_Stop_Fn, void *);
2197            extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2198            extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2199            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2200            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2201            _Unwind_Context
2202                                                               *);
2203            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2204            extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2205            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2206            _Unwind_Exception
2207                                                              *);
2208            extern void _Unwind_Resume(struct _Unwind_Exception *);
2209            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2210            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2211            extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2212            extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2213            extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2214                                                    _Unwind_Stop_Fn, void *);
2215            extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2216            extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2217            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2218            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2219            _Unwind_Context
2220                                                               *);
2221            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2222            extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2223            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2224            _Unwind_Exception
2225                                                              *);
2226            extern void _Unwind_Resume(struct _Unwind_Exception *);
2227            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2228            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2229            extern void _Unwind_DeleteException(struct _Unwind_Exception *);
```

```
2230            extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2231            extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2232                                                    _Unwind_Stop_Fn, void *);
2233            extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2234            extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2235            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2236            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(struct
2237            _Unwind_Context
2238                                                               *);
2239            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2240            extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2241            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2242            _Unwind_Exception
2243                                                              *);
2244            extern void _Unwind_Resume(struct _Unwind_Exception *);
2245            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2246            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2247            extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2248            extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2249            extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2250                                                    _Unwind_Stop_Fn, void *);
2251            extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2252            extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2253            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2254            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2255            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2256            extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2257            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2258            _Unwind_Exception
2259                                                              *);
2260            extern void _Unwind_Resume(struct _Unwind_Exception *);
2261            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2262            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2263            extern void _Unwind_DeleteException(struct _Unwind_Exception *);
2264            extern fde *_Unwind_Find_FDE(void *, struct dwarf_eh_base *);
2265            extern _Unwind_Ptr _Unwind_ForcedUnwind(struct _Unwind_Exception *,
2266                                                    _Unwind_Stop_Fn, void *);
2267            extern _Unwind_Ptr _Unwind_GetDataRelBase(struct _Unwind_Context *);
2268            extern _Unwind_Word _Unwind_GetGR(struct _Unwind_Context *, int);
2269            extern _Unwind_Ptr _Unwind_GetIP(struct _Unwind_Context *);
2270            extern _Unwind_Ptr _Unwind_GetLanguageSpecificData(void);
2271            extern _Unwind_Ptr _Unwind_GetRegionStart(struct _Unwind_Context *);
2272            extern _Unwind_Ptr _Unwind_GetTextRelBase(struct _Unwind_Context *);
2273            extern _Unwind_Reason_Code _Unwind_RaiseException(struct
2274            _Unwind_Exception
2275                                                              *);
2276            extern void _Unwind_Resume(struct _Unwind_Exception *);
2277            extern void _Unwind_SetGR(struct _Unwind_Context *, int, u_int64_t);
2278            extern void _Unwind_SetIP(struct _Unwind_Context *, _Unwind_Ptr);
2279            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2280            *);
2281            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2282            *);
2283            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2284            *);
2285            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2286            *);
2287            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2288            *);
2289            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2290            *);
2291            extern _Unwind_Reason_Code _Unwind_Backtrace(_Unwind_Trace_Fn, void
2292            *);
2293            extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
```

```
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(struct _Unwind_Context *);
extern _Unwind_Reason_Code _Unwind_GetCFA(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 *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern _Unwind_Reason_Code _Unwind_Resume_or_Rethrow(struct

_Unwind_Exception *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern void *_Unwind_FindEnclosingFunction(void *);
extern _Unwind_Word _Unwind_GetBSP(struct _Unwind_Context *);
```

11.10 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 11.8 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

```
_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 parameteres 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*.

_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 returnd:

_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.

_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 *index*ed 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*

11.11 Interfaces for libdl

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

Table 11-33 libdl Definition

Library:	libdl
SONAME:	libdl.so.2

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

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

11.11.1 Dynamic Loader

11.11.1.1 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_2.2) [LSB]	dlclose(GLIBC_2.2) [SUSv3]	dlerror(GLIBC_2.2) [SUSv3]	dlopen(GLIBC_2.2) [LSB]

dlsym(GLIBC_2.2) [LSB]			

11.12 Data Definitions for libdl

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

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

This specification uses the 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.

11.12.1 dlfcn.h

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

11.13 Interfaces for libcrypt

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

Table 11-35 libcrypt Definition

Library:	libcrypt
SONAME:	libcrypt.so.1

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

[SUSv3] ISO POSIX (2003)

11.13.1 Encryption

11.13.1.1 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

crypt(GLIBC_2.2) [SUSv3]	encrypt(GLIBC_2.2) [SUSv3]	setkey(GLIBC_2.2) [SUSv3]	

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 12-1 libz Definition

Library:	libz
SONAME:	libz.so.1

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, 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 gzprintf(gzFile, const char *, ...);
extern int gzputc(gzFile, int);
extern int gzputs(gzFile, const char *);
extern int gzrewind(gzFile);
extern z_off_t gzseek(gzFile, z_off_t, int);
extern int gzsetparams(gzFile, int, int);
extern z_off_t gztell(gzFile);
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);
```

12.3 Interfaces for libncurses

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

Table 12-2 libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

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,
                   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 deleteln(void);
extern WINDOW *derwin(WINDOW *, int, int, int, int);
extern int doupdate(void);
extern 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);
```

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

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

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

```
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 winnstr(WINDOW *, char *, int);
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 wredrawln(WINDOW *, int, int);
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 char *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)
    );
```

```
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 12-3 libutil Definition

Library:	libutil
SONAME:	libutil.so.1

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 12-4 libutil - Utility Functions Function Interfaces

forkpty(GLIBC_2.2) [LSB]	login(GLIBC_2.2) [LSB]	login_tty(GLIBC_2.2) [LSB]	logout(GLIBC_2.2) [LSB]
logwtmp(GLIBC_2.2) [LSB]	openpty(GLIBC_2.2) [LSB]		

V Package Format and Installation

13 Software Installation

13.1 Package Dependencies

The LSB runtime environment shall provde the following dependencies.

lsb-core-s390x

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`*`-s390x`.

13.2 Package Architecture Considerations

All packages must specify an architecture of `s390x`. A LSB runtime environment must accept an architecture of `s390` even if the native architecture is different.

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

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

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

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 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.