

**Linux Standard Base Core Module
Specification for IA64 2.0.1**

Linux Standard Base Core Module Specification for IA64 2.0.1

Copyright © 2004 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.

OpenGL is a registered trademark of Silicon Graphics, Inc.

Specification Introduction

Specification Introduction

Table of Contents

Foreword	i
Introduction	ii
I. Introductory Elements	3
1. Scope	1
1.1. General	1
1.2. Module Specific Scope	1
2. Normative References	2
3. Requirements	6
3.1. Relevant Libraries	6
3.2. LSB Implementation Conformance	6
3.3. LSB Application Conformance	7
4. Definitions	8
5. Terminology	9
6. Documentation Conventions	10

List of Tables

2-1. Normative References	2
3-1. Standard Library Names	6

Foreword

1 | This is version 2.0.1 of the Linux Standard Base Core Module Specification for IA64. An implementation of this
2 | version of the specification may not claim to be an implementation of the Linux Standard Base unless it has
3 | successfully completed the compliance process as defined by the Free Standards Group.

Introduction

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

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

I. Introductory Elements

Chapter 1. Scope

1.1. General

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

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

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

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

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

1.2. Module Specific Scope

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

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

Chapter 2. Normative References

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

4 **Table 2-1. Normative References**

System V Application Binary Interface—DRAFT—17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://www.eagereon.com/dwarf/dwarf-2.0.0.pdf
Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEEE Standard 754 for Binary Floating-Point Arithmetic	http://www.ieee.org/
System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspeecs/gabi41.pdf
Intel® Itanium™ Processor-specific Application Binary Interface	http://refspecs.freestandards.org/cfi/IA64-SysV-psABI.pdf
Itanium™ Software Conventions & Runtime Architecture Guide	http://refspecs.freestandards.org/IA64conventions.pdf
Itanium™ Architecture Software Developer's Manual Volume 1: Application Architecture	http://refspecs.freestandards.org/IA64-softdevman-vol1.pdf
Itanium™ Architecture Software Developer's Manual Volume 2: System Architecture	http://refspecs.freestandards.org/IA64-softdevman-vol2.pdf
Itanium™ Architecture Software Developer's Manual Volume 3: Instruction Set Reference	http://refspecs.freestandards.org/IA64-softdevman-vol3.pdf
IA-64 Processor Reference: Intel® Itanium™ Processor Reference Manual for Software Development	http://refspecs.freestandards.org/IA64-softdevman-vol4.pdf
ISO/IEC 9899: 1999, Programming Languages—C	
Linux Assigned Names And Numbers Authority	http://www.lanana.org/
Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
LI18NIX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NIX-2000-amd4.htm
Linux Standard Base	http://www.linuxbase.org/spec/
OSF RFC 86.0	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.t

		xt
RFC 1833: Binding Protocols for ONC RPC Version 2		http://www.ietf.org/rfc/rfc1833.txt
RFC 1952: GZIP file format specification version 4.3		http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format		http://www.ietf.org/rfc/rfc2440.txt
CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018		http://www.opengroup.org/publications/catalog/un.htm
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
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
ISO/IEC 9945:2003 Portable Operating System(POSIX)and The Single UNIX® Specification(SUS) V3		http://www.unix.org/version3/
System V Interface Definition, Issue 3 (ISBN 0201566524)		
System V Interface Definition, Fourth Edition		
zlib 1.2 Manual		http://www.gzip.org/zlib/
Name	Title	URL
DWARF Debugging Information Format	DWARF Debugging Information Format, Revision 2.0.0 (July 27, 1993)	http://www.eagercon.com/dwarf/dwarf-2.0.0.pdf
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
IEEE Std 754-1985	IEEE Standard 754 for Binary Floating-Point Arithmetic	http://www.ieee.org/
Intel® Itanium™ Processor-specific Application Binary Interface	Intel® Itanium™ Processor-specific Application Binary Interface	http://refspecs.freestandards.org/elf/IA64-SysV-psABI.pdf
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
ISO POSIX (2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information	http://www.unix.org/version3/

	<p>technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces</p> <p>ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities</p> <p>ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale</p>	
Itanium™ Architecture Software Developer's Manual Volume 1	Itanium™ Architecture Software Developer's Manual Volume 1: Application Architecture	http://refspecs.freestandards.org/IA64-softdevman-vol1.pdf
Itanium™ Architecture Software Developer's Manual Volume 2	Itanium™ Architecture Software Developer's Manual Volume 2: System Architecture	http://refspecs.freestandards.org/IA64-softdevman-vol2.pdf
Itanium™ Architecture Software Developer's Manual Volume 3	Itanium™ Architecture Software Developer's Manual Volume 3: Instruction Set Reference	http://refspecs.freestandards.org/IA64-softdevman-vol3.pdf
Itanium™ Architecture Software Developer's Manual Volume 4	IA-64 Processor Reference: Intel® Itanium™ Processor Reference Manual for Software Development	http://refspecs.freestandards.org/IA64-softdevman-vol4.pdf
Itanium™ Software Conventions and Runtime Guide	Itanium™ Software Conventions & Runtime Architecture Guide, September 2000	http://refspecs.freestandards.org/IA64conventions.pdf
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
Li18nux Globalization Specification	LI18NUNIX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.li18nux.org/docs/html/LI18NUNIX-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices.txt
PAM	Open Software Foundation, Request For Comments: 86.0, October 1995, V. Samar & R.Schemers (SunSoft)	http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt

RFC 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
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 Command and Utilities	The Single UNIX® Specification(SUS) Version 2, Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)	http://www.opengroup.org/publications/catalog/un.htm
SVID Issue 3	American Telephone and Telegraph Company, System V Interface Definition, Issue 3 ; Morristown, NJ, UNIX Press, 1989.(ISBN 0201566524)	
SVID Issue 4	System V Interface Definition, Fourth Edition	
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.caldera.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.caldera.com/developers/gabi/2003-12-17/contents.html
this specification	Linux Standard Base	http://www.linuxbase.org/spec/
X/Open Curses	CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), plus Corrigendum U018	http://www.opengroup.org/publications/catalog/un.htm
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

Chapter 3. Requirements

3.1. Relevant Libraries

1 The libraries listed in Table 3-1 shall be available on IA64 Linux Standard Base systems, with the specified runtime
2 names. These names override or supplement the names specified in the generic LSB specification. The specified
3 program interpreter, referred to as proginterp in this table, shall be used to load the shared libraries specified by
4 DT_NEEDED entries at run time.

5 **Table 3-1. Standard Library Names**

Library	Runtime Name
libm	libm.so.6.1
libc	libc.so.6.1
proginterp	/lib/ld-lsb-ia64.so.2
libpthread	libpthread.so.0
libdl	libdl.so.2
libcrypt	libcrypt.so.1
libgcc_s	libgcc_s.so.1
libz	libz.so.1
libncurses	libncurses.so.5
libutil	libutil.so.1
libe	libe.so.6.1
libpthread	libpthread.so.0
proginterp	/lib/ld-lsb-ia64.so.2
libgcc_s	libgcc_s.so.1

6
7 These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

3.2. LSB Implementation Conformance

8 A conforming implementation shall satisfy the following requirements:

- 9 • The implementation shall implement fully the architecture described in the hardware manual for the target
10 processor architecture.
- 11 • The implementation shall be capable of executing compiled applications having the format and using the system
12 interfaces described in this document.

- 13 • The implementation shall provide libraries containing the interfaces specified by this document, and shall provide a
14 dynamic linking mechanism that allows these interfaces to be attached to applications at runtime. All the interfaces
15 shall behave as specified in this document.
- 16 • The map of virtual memory provided by the implementation shall conform to the requirements of this document.
- 17 • The implementation's low-level behavior with respect to function call linkage, system traps, signals, and other such
18 activities shall conform to the formats described in this document.
- 19 • The implementation shall provide all of the mandatory interfaces in their entirety.
- 20 • The implementation may provide one or more of the optional interfaces. Each optional interface that is provided
21 shall be provided in its entirety. The product documentation shall state which optional interfaces are provided.
- 22 • The implementation shall provide all files and utilities specified as part of this document in the format defined here
23 and in other referenced documents. All commands and utilities shall behave as required by this document. The
24 implementation shall also provide all mandatory components of an application's runtime environment that are
25 included or referenced in this document.
- 26 • The implementation, when provided with standard data formats and values at a named interface, shall provide the
27 behavior defined for those values and data formats at that interface. However, a conforming implementation may
28 consist of components which are separately packaged and/or sold. For example, a vendor of a conforming
29 implementation might sell the hardware, operating system, and windowing system as separately packaged items.
- 30 • The implementation may provide additional interfaces with different names. It may also provide additional
31 behavior corresponding to data values outside the standard ranges, for standard named interfaces.

3.3. LSB Application Conformance

32 | ~~Ann~~A conforming application shall satisfy the following requirements:

- 33 • Its executable files are either shell scripts or object files in the format defined for the Object File Format system
34 interface.
- 35 • Its object files participate in dynamic linking as defined in the Program Loading and Linking System interface.
- 36 • It employs only the instructions, traps, and other low-level facilities defined in the Low-Level System interface as
37 being for use by applications.
- 38 • If it requires any optional interface defined in this document in order to be installed or to execute successfully, the
39 requirement for that optional interface is stated in the application's documentation.
- 40 • It does not use any interface or data format that is not required to be provided by a conforming implementation,
41 unless:
 - 42 • If such an interface or data format is supplied by another application through direct invocation of that application
43 during execution, that application is in turn an LSB conforming application.
 - 44 • The use of that interface or data format, as well as its source, is identified in the documentation of the application.
- 45 • It shall not use any values for a named interface that are reserved for vendor extensions.

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

Chapter 4. Definitions

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

3 can

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

5 cannot

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

7 may

8 is permitted; is allowed; is permissible

9 need not

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

11 shall

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

13 shall not

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

15 should

16 it is recommended that; ought to

17 should not

18 it is not recommended that; ought not to

Chapter 5. Terminology

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

2 archLSB

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

5 Binary Standard

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

7 gLSB

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

10 implementation-defined

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

16 Shell Script

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

19 Source Standard

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

21 undefined

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

27 unspecified

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

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

Chapter 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) [1]

refers to the interface named `forkpty` with symbol version `GLIBC_2.0` that is defined in the first of the listed references below the table.

ELF Specification

2

3 **ELF Specification**

Table of Contents

I. Low Level System Information	17
1. Machine Interface.....	1
1.1. Processor Architecture.....	1
1.2. Data Representation.....	1
1.2.1. Byte Ordering.....	2
1.2.2. Fundamental Types.....	2
1.2.3. Aggregates and Unions.....	3
1.2.4. Bit Fields.....	4
2. Function Calling Sequence.....	6
2.1. CPU Registers.....	6
2.2. Floating Point Registers.....	6
2.3. Stack Frame.....	6
2.4. Arguments.....	6
2.4.1. Integral/Pointer.....	6
2.4.2. Floating Point.....	6
2.4.3. Struct and Union Point.....	6
2.4.4. Variable Arguments.....	6
2.5. Return Values.....	6
2.5.1. Void.....	7
2.5.2. Integral/Pointer.....	7
2.5.3. Floating Point.....	7
2.5.4. Struct and Union.....	7
3. Operating System Interface.....	8
3.1. Processor Execution Mode.....	8
3.2. Exception Interface.....	8
3.2.1. Hardware Exception Types.....	8
3.2.2. Software Trap Types.....	8
3.2.3. Debugging Support.....	8
3.2.4. Process Startup.....	8
3.3. Signal Delivery.....	8
3.3.1. Signal Handler Interface.....	8
4. Process Initialization.....	9
4.1. Special Registers.....	9
4.2. Process Stack (on entry).....	9
4.3. Auxiliary Vector.....	9
4.4. Environment.....	11
5. Coding Examples.....	12
5.1. Code Model Overview/Architecture Constraints.....	12
5.2. Position-Independent Function Prologue.....	12
5.3. Data Objects.....	12
5.3.1. Absolute Load & Store.....	12
5.3.2. Position Relative Load & Store.....	12
5.4. Function Calls.....	12

5.4.1. Absolute Direct Function Call.....	12
5.4.2. Absolute Indirect Function Call	13
5.4.3. Position-Independent Direct Function Call	13
5.4.4. Position-Independent Indirect Function Call.....	13
5.5. Branching.....	13
5.5.1. Branch Instruction	13
5.5.2. Absolute switch() code.....	13
5.5.3. Position-Independent switch() code	13
6. C Stack Frame	14
6.1. Variable Argument List	14
6.2. Dynamic Allocation of Stack Space	14
7. Debug Information	15
II. Object Format.....	16
8. ELF Header	17
8.1. Machine Information	17
8.1.1. File Class.....	17
8.1.2. Data Encoding	17
8.1.3. OS Identification	17
8.1.4. Processor Identification.....	17
8.1.5. Processor Specific Flags.....	17
9. Sections	18
9.1. Special Sections	18
9.2. Linux Special Sections.....	19
9.3. Section Types.....	20
9.4. Section Attribute Flags	20
9.5. Special Section Types.....	20
10. Symbol Table	21
11. Relocation	22
11.1. Relocation Types	22
III. Program Loading and Dynamic Linking	23
12. Program Header.....	24
12.1. Types	24
12.2. Flags.....	24
13. Program Loading.....	25
14. Dynamic Linking.....	26
14.1. Dynamic Entries	26
14.1.1. ELF Dynamic Entries.....	26
14.1.2. Additional Dynamic Entries.....	26
14.2. Global Offset Table	26
14.3. Shared Object Dependencies	26
14.4. Function Addresses.....	26
14.5. Procedure Linkage Table	26
14.6. Initialization and Termination Functions.....	26

List of Tables

1-1. Scalar Types	2
8-1. Additional Processor-Specific Flags	17
9-1. ELF Special Sections.....	18
9-2. Additional Special Sections.....	19

List of Figures

1-1. Structure Smaller Than A Word.....	3
1-2. No Padding.....	3
1-3. Internal and Tail Padding	4
1-4. Bit-Field Ranges.....	4

I. Low Level System Information

Chapter 1. Machine Interface

1.1. Processor Architecture

1 The Architecture is specified by the following documents

- 2 • ~~Itanium™ Architecture Software Developer's Manual Volume 1: Application Architecture~~
- 3 • ~~Itanium™ Architecture Software Developer's Manual Volume 2: System Architecture~~
- 4 • ~~Itanium™ Architecture Software Developer's Manual Volume 3: Instruction Set Reference~~
- 5 • ~~IA-64 Processor Reference: Intel® Itanium™ Processor Reference Manual for Software Development~~ Itanium™
6 Architecture Software Developer's Manual Volume 4
- 7 • ~~Itanium™ Software Conventions & Runtime Architecture Guide~~
- 8 • Intel® Itanium™ Processor-specific Application Binary Interface

9 Only the features of the processor instruction set may be assumed to be present. An application is responsible for
10 determining if any additional instruction set features are available before using those additional features. If a feature is
11 not present, then the application may not use it.

12 Only instructions which do not require elevated privileges may be used.

13 Applications may not make system calls directly. The interfaces in the C library must be used instead.

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

16 Applications conforming to this specification must provide feedback to the user if a feature that is required for correct
17 execution of the application is not present. Applications conforming to this specification should attempt to execute in
18 a diminished capacity if a required feature is not present.

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

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

1.2. Data Representation

25 See Itanium™ Software Conventions & Runtime ~~Architecture Guide~~ Chapter 4.

26 Within this specification, the term `byte` refers to an 8-bit object, the term `halfword` refers to a 16-bit object, the term
27 `word` refers to a 32-bit object, the term `doubleword` refers to a 64-bit object, and the term `quadword` refers to a
28 128-bit object. Although the architecture also supports 120-bit addressable objects, this specification does not require
29 LSB-conforming implementations to provide support for these objects.

1.2.1. Byte Ordering

30 LSB-conforming applications shall use little-endian byte ordering. LSB-conforming implementations may support
31 big-endian applications.

1.2.2. Fundamental Types

32 Table 2-1 describes how fundamental C language data types shall be represented:

33 **Table 1-1. Scalar Types**

Type	C	sizeof	Alignment (bytes)	Notes
Integral	char	1	1	
	signed char			
	unsigned char			
	short	2	2	
	signed short			
	unsigned short			
	int	4	4	
	signed int			
	unsigned int			
	long	8	8	
	signed long			
	unsigned long			
	long long	8	8	See Note Below
signed long long				
unsigned long long				
Pointer	<i>any-type</i> *	8	8	
	<i>any-type</i> (*)()			
Floating-Point	float	4	4	
	double	8	8	
	long double	16	16	

34
35 Support for the `long long` data type is dependent on support for ISO9899:1999 C language. This standard is not
36 required for LSB-conformance, but this data type is important when developing applications for the architecture.
37 The GNU Compiler Collection (gcc) includes support for `long long` of ISO9899:1999.

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

1.2.3. Aggregates and Unions

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

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

48 A conforming application shall not read padding.

49 **Figure 1-1. Structure Smaller Than A Word**

<pre> struct { char c; } </pre>				
Byte aligned, sizeof is 1				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 20%; padding: 5px;">Offset</th> <th style="padding: 5px;">Byte 0</th> </tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 5px;">0</td> <td style="text-align: center; padding: 5px;">c^0</td> </tr> </tbody> </table>	Offset	Byte 0	0	c^0
Offset	Byte 0			
0	c^0			

52 **Figure 1-2. No Padding**

<pre> struct { char c; char d; short s; int i; long l; } </pre>																									
Doubleword Aligned, sizeof is 16																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%; padding: 5px;">Offset</th> <th style="width: 15%; padding: 5px;">Byte 3</th> <th style="width: 15%; padding: 5px;">Byte 2</th> <th style="width: 15%; padding: 5px;">Byte 1</th> <th style="width: 15%; padding: 5px;">Byte 0</th> </tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 5px;">0</td> <td colspan="2" style="text-align: center; padding: 5px;">s^2</td> <td style="text-align: center; padding: 5px;">d^1</td> <td style="text-align: center; padding: 5px;">c^0</td> </tr> <tr> <td style="text-align: center; padding: 5px;">4</td> <td colspan="4" style="text-align: center; padding: 5px;">i^0</td> </tr> <tr> <td style="text-align: center; padding: 5px;">8</td> <td colspan="4" style="text-align: center; padding: 5px;">l^0</td> </tr> <tr> <td style="text-align: center; padding: 5px;">12</td> <td colspan="4"></td> </tr> </tbody> </table>	Offset	Byte 3	Byte 2	Byte 1	Byte 0	0	s^2		d^1	c^0	4	i^0				8	l^0				12				
Offset	Byte 3	Byte 2	Byte 1	Byte 0																					
0	s^2		d^1	c^0																					
4	i^0																								
8	l^0																								
12																									

55 **Figure 1-3. Internal and Tail Padding**

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

1.2.4. Bit Fields

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

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

- 62 • Bit-fields are allocated from right to left (least to most significant).
- 63 • A bit-field must entirely reside in a storage unit for its appropriate type. A bit field shall never cross its unit
64 boundary.
- 65 • Bit-fields may share a storage unit with other `struct/union` members, including members that are not bit fields.
66 Such other `struct/union` members shall occupy different parts of the storage unit.
- 67 • The type of unnamed bit-fields shall not affect the alignment of a structure or union, although individual bit-field
68 member offsets shall obey the alignment constraints.

69 **Figure 1-4. Bit-Field Ranges**

Bit-field Type	Width	Range
signed char char unsigned char	1 to 8	-2 ⁻¹ to 2 ⁻¹ -1 0 to 2-1 0 to 2-1
signed short short	1 to 16	-2 ⁻¹ to 2 ⁻¹ -1 0 to 2-1

Bit-field Type	Width	Range
unsigned short		0 to $2^{16}-1$
signed int int unsigned int	1 to 32	-2^{31} to $2^{31}-1$ 0 to $2^{32}-1$ 0 to $2^{32}-1$
signed long long unsigned long	1 to 64	-2^{63} to $2^{63}-1$ 0 to $2^{64}-1$ 0 to $2^{64}-1$

70

Chapter 2. Function Calling Sequence

1 LSB-conforming applications shall use the procedure linkage and function calling sequence as defined in Chapter 8.4
2 of the Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide.

2.1. CPU Registers

3 The CPU general and other registers are as defined in the Itanium™ Architecture Software Developer's Manual
4 Volume 1: ~~Application Architecture~~ Section 3.1.

2.2. Floating Point Registers

5 The floating point registers are as defined in the Itanium™ Architecture Software Developer's Manual Volume 1:
6 ~~Application Architecture~~ Section 3.1.

2.3. Stack Frame

7 The stackframe layout is as described in the Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide
8 Chapter 8.4.

2.4. Arguments

9 The procedure argument passing mechanism is as described in the Itanium™ Software Conventions &and Runtime
10 ~~Architecture~~ Guide Chapter 8.5.

2.4.1. Integral/Pointer

11 See Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide Chapter 8.5.

2.4.2. Floating Point

12 See Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide Chapter 8.5.

2.4.3. Struct and Union Point

13 See Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide Chapter 8.5.

2.4.4. Variable Arguments

14 See Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide Chapter 8.5.4.

2.5. Return Values

15 See Itanium™ Software Conventions &and Runtime ~~Architecture~~ Guide Chapter 8.6.

2.5.1. Void

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

2.5.2. Integral/Pointer

17 | See Itanium™ Software Conventions & Runtime Architecture Guide Chapter 8.6.

2.5.3. Floating Point

18 | See Itanium™ Software Conventions & Runtime Architecture Guide Chapter 8.6.

2.5.4. Struct and Union

19 | See Itanium™ Software Conventions & Runtime Architecture Guide Chapter 8.6 (aggregate return values).

20 Depending on the size (including any padding), aggregate data types may be passed in one or more general registers,
21 or in memory.

Chapter 3. Operating System Interface

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

3.1. Processor Execution Mode

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

3.2. Exception Interface

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

3.2.1. Hardware Exception Types

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

3.2.2. Software Trap Types

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

3.2.3. Debugging Support

8 See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.4.

3.2.4. Process Startup

9 See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.5.

3.3. Signal Delivery

10 See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.2.

3.3.1. Signal Handler Interface

11 See Intel® Itanium™ Processor-specific Application Binary Interface, section 3.3.3.

Chapter 4. Process Initialization

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

4.1. Special Registers

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

4.2. Process Stack (on entry)

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

4.3. Auxiliary Vector

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

```
13 typedef struct  
14 {  
15     long int a_type;           /* Entry type */  
16     union  
17     {  
18         long int a_val;       /* Integer value */  
19         void *a_ptr;         /* Pointer value */  
20         void (*a_fcn) (void); /* Function pointer value */  
21     } a_un;  
22 } auxv_t;
```

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

24 The a_type field shall contain one of the following values:

25 AT_NULL

26 The last entry in the array has type AT_NULL. The value in a_un is undefined.

27 AT_IGNORE

28 The value in a_un is undefined, and should be ignored.

29	AT_EXECFD
30	File descriptor of program
31	AT_PHDR
32	Program headers for program
33	AT_PHENT
34	Size of program header entry
35	AT_PHNUM
36	Number of program headers
37	AT_PAGESZ
38	System page size
39	AT_BASE
40	Base address of interpreter
41	AT_FLAGS
42	Flags
43	AT_ENTRY
44	Entry point of program
45	AT_NOTELF
46	Program is not ELF
47	AT_UID
48	Real uid
49	AT_EUID
50	Effective uid
51	AT_GID
52	Real gid
53	AT_EGID
54	Effective gid
55	AT_CLKTCK
56	Frequency of times()
57	AT_PLATFORM
58	String identifying platform.

- 59 AT_HWCAP
60 Machine dependent hints about processor capabilities.
- 61 AT_FPUCW
62 Used FPU control word
- 63 AT_DCACHEBSIZE
64 Data cache block size
- 65 AT_ICACHEBSIZE
66 Instruction cache block size
- 67 AT_UCACHEBSIZE
68 Unified cache block size
- 69 The auxiliary vector is intended for passing information from the operating system to the program interpreter.

4.4. Environment

- 70 Although a pointer to the environment vector should be available as a third argument to the main entry point,
71 conforming applications should use `getenv` to access the environment. (See ~~ISO/IEC 9945~~: POSIX (2003-Portable
72 ~~Operating System(POSIX)and The Single UNIX® Specification(SUS) V3~~), Section `exec`).

Chapter 5. Coding Examples

1 LSB-conforming applications may implement fundamental operations using the Coding Examples as shown below.
2 Sample code sequences and coding conventions can be found in Itanium™ Software Conventions & Runtime
3 Architecture Guide, Chapter 9.

5.1. Code Model Overview/Architecture Constraints

4 As defined in Intel® Itanium™ Processor-specific Application Binary Interface, relocatable files, executable files,
5 and shared object files that are supplied as part of an application must use Position Independent Code, as described in
6 Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 12.

5.2. Position-Independent Function Prologue

7 See Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 8.4.

5.3. Data Objects

8 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 5.3.4, and Itanium™ Software
9 Conventions & Runtime Architecture Guide, Chapter 12.3.

5.3.1. Absolute Load & Store

10 Conforming applications shall not use absolute addressing.

5.3.2. Position Relative Load & Store

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

5.4. Function Calls

12 See Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 8.4.

13 Four types of procedure call are defined in Itanium™ Software Conventions & Runtime Architecture Guide,
14 Chapter 8.3. Although special calling conventions are permitted, provided that the compiler and runtime library agree
15 on these conventions, none are defined for this standard. Consequently, no application shall depend on a type of
16 procedure call other than Direct Calls, Direct Dynamically Linked Calls, or Indirect Calls, as defined in Itanium™
17 Software Conventions & Runtime Architecture Guide, Chapter 8.3.

5.4.1. Absolute Direct Function Call

18 Conforming applications shall not use absolute addressing.

5.4.2. Absolute Indirect Function Call

19 Conforming applications shall not use absolute addressing.

5.4.3. Position-Independent Direct Function Call

20 See Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 8.4.1.

5.4.4. Position-Independent Indirect Function Call

21 See Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 8.4.2.

5.5. Branching

22 Branching is described in ~~IA-64 Processor Reference: Intel® Itanium™ Processor Reference Manual for Software~~
23 ~~Development~~ Itanium™ Architecture Software Developer's Manual Volume 4, Chapter 4.5.

5.5.1. Branch Instruction

24 See ~~IA-64 Processor Reference: Intel® Itanium™ Processor Reference Manual for Software Development~~ Itanium™
25 Architecture Software Developer's Manual Volume 4, Chapter 4.5.

5.5.2. Absolute switch() code

26 Conforming applications shall not use absolute addressing.

5.5.3. Position-Independent switch() code

27 Where there are several possible targets for a branch, the compiler may use a number of different code generation
28 strategies. See Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 9.1.7.

Chapter 6. C Stack Frame

6.1. Variable Argument List

- 1 | See Itanium™ Software Conventions & Runtime Architecture Guide, Chapter 8.5.2, and 8.5.4.

6.2. Dynamic Allocation of Stack Space

- 2 | The C library `alloca` function should be used to dynamically allocate stack space.

Chapter 7. Debug Information

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

II. Object Format

- 2 LSB-conforming implementations shall support an object file , called Executable and Linking Format (ELF) as
- 3 | defined by the System V ~~Application Binary Interface, Edition 4.~~ABI, Intel® Itanium™ Processor-specific
- 4 Application Binary Interface and as supplemented by the Linux Standard Base Specification and this document.

Chapter 8. ELF Header

8.1. Machine Information

1 LSB-conforming applications shall use the Machine Information as defined in Intel® Itanium™ Processor-specific
2 Application Binary Interface, Chapter 4. Implementations shall support the LP64 model. It is unspecified whether or
3 not the ILP32 model shall also be supported.

8.1.1. File Class

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

8.1.2. Data Encoding

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

8.1.3. OS Identification

8 The OS Identification field `e_ident[EI_OSABI]` shall contain the value `ELFOSABI_LINUX`.

8.1.4. Processor Identification

9 The processor identification value held in `e_machine` shall contain the value `EM_IA_64`.

8.1.5. Processor Specific Flags

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

12 The following additional processor-specific flags are defined:

13 **Table 8-1. Additional Processor-Specific Flags**

Name	Value
EF_IA_64_LINUX_EXECUTABLE_STACK	0x00000001

14 EF_IA_64_LINUX_EXECUTABLE_STACK

15 The stack and heap sections are executable. If this flag is not set, code can not be executed from the stack or heap.
16

Chapter 9. Sections

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

9.1. Special Sections

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

4 **Table 9-1. ELF Special Sections**

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

5
6 .got

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

9 .IA_64.archext

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

13 .IA_64.pltoff

14 This section holds local function descriptor entries.

15 `.IA_64.unwind`
 16 This section holds the unwind function table. The contents are described in the Intel (r) Itanium (tm) Processor
 17 Specific ABI.

18 `.IA_64.unwind_info`
 19 This section holds stack unwind and and exception handling information. The exception handling information is
 20 programming language specific, and is unspecified.

21 `.plt`
 22 This section holds the Procedure Linkage Table.

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

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

32 `.sdata1`
 33 See `.sdata`.

9.2. Linux Special Sections

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

35 **Table 9-2. Additional Special Sections**

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

37 `.opd`
 38 This section holds function descriptors

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

42 .rel.IA_64.pltoff

43 This section holds relocation information, as described in `Relocation`. These relocations are applied to
44 the .IA_64.pltoff section.

9.3. Section Types

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

9.4. Section Attribute Flags

50 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.2.2.

9.5. Special Section Types

51 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.2.3.

Chapter 10. Symbol Table

1 If an executable file contains a reference to a function defined in one of its associated shared objects, the symbol table
2 section for that file shall contain an entry for that symbol. The `st_shndx` member of that symbol table entry contains
3 `SHN_UNDEF`. This signals to the dynamic linker that the symbol definition for that function is not contained in the
4 executable file itself. If that symbol has been allocated a procedure linkage table entry in the executable file, and the
5 `st_value` member for that symbol table entry is non-zero, the value shall contain the virtual address of the first
6 instruction of that procedure linkage table entry. Otherwise, the `st_value` member contains zero. This procedure
7 linkage table entry address is used by the dynamic linker in resolving references to the address of the function.

8 Need to add something here about `st_info` and `st_other` ...

Chapter 11. Relocation

- 1 LSB-conforming applications shall use Relocations as defined in Intel® Itanium™ Processor-specific Application
- 2 Binary Interface, Chapter 4.3.

11.1. Relocation Types

- 3 See Intel® Itanium™ Processor-specific Application Binary Interface, Chapter 4.3.

III. Program Loading and Dynamic Linking

- 2 LSB-conforming implementations shall support the object file information and system actions that create running
3 programs as specified in the System V ~~Application Binary Interface, Edition 4.1~~ABI, Intel® Itanium™
4 Processor-specific Application Binary Interface and as supplemented by the Linux Standard Base Specification and
5 this document.

Chapter 12. Program Header

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

12.1. Types

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

12.2. Flags

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

Chapter 13. Program Loading

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

Chapter 14. Dynamic Linking

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

14.1. Dynamic Entries

14.1.1. ELF Dynamic Entries

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

4 DT_PLTGOT

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

14.1.2. Additional Dynamic Entries

6 The following dynamic entries are defined here.

7 DT_RELACOUNT

8 The number of relative relocations in `.rela.dyn`

14.2. Global Offset Table

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

14.3. Shared Object Dependencies

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

14.4. Function Addresses

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

14.5. Procedure Linkage Table

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

14.6. Initialization and Termination Functions

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

Linux Standard Base Specification

2

3 **Linux Standard Base Specification**

Table of Contents

I. Base Libraries	33
1. Libraries	1
1.1. Program Interpreter/Dynamic Linker	1
1.2. Interfaces for libc	1
1.2.1. RPC	1
1.2.1.1. Interfaces for RPC	1
1.2.2. System Calls	3
1.2.2.1. Interfaces for System Calls	3
1.2.3. Standard I/O	5
1.2.3.1. Interfaces for Standard I/O	5
1.2.4. Signal Handling	7
1.2.4.1. Interfaces for Signal Handling	7
1.2.5. Localization Functions	8
1.2.5.1. Interfaces for Localization Functions	8
1.2.6. Socket Interface	9
1.2.6.1. Interfaces for Socket Interface	9
1.2.7. Wide Characters	10
1.2.7.1. Interfaces for Wide Characters	10
1.2.8. String Functions	12
1.2.8.1. Interfaces for String Functions	12
1.2.9. IPC Functions	13
1.2.9.1. Interfaces for IPC Functions	13
1.2.10. Regular Expressions	14
1.2.10.1. Interfaces for Regular Expressions	14
1.2.11. Character Type Functions	15
1.2.11.1. Interfaces for Character Type Functions	15
1.2.12. Time Manipulation	16
1.2.12.1. Interfaces for Time Manipulation	16
1.2.13. Terminal Interface Functions	17
1.2.13.1. Interfaces for Terminal Interface Functions	17
1.2.14. System Database Interface	17
1.2.14.1. Interfaces for System Database Interface	17
1.2.15. Language Support	18
1.2.15.1. Interfaces for Language Support	18
1.2.16. Large File Support	19
1.2.16.1. Interfaces for Large File Support	19
1.2.17. Standard Library	19
1.2.17.1. Interfaces for Standard Library	19
1.3. Data Definitions for libc	23
1.3.1. errno.h	23
1.3.2. inttypes.h	23
1.3.3. limits.h	23
1.3.4. setjmp.h	23

1.3.5. signal.h	23
1.3.6. stddef.h	24
1.3.7. sys/ioctl.h	25
1.3.8. sys/ipc.h	25
1.3.9. sys/mman.h	25
1.3.10. sys/msg.h	25
1.3.11. sys/sem.h	26
1.3.12. sys/shm.h	26
1.3.13. sys/socket.h	26
1.3.14. sys/stat.h	26
1.3.15. sys/statvfs.h	27
1.3.16. sys/types.h	28
1.3.17. termios.h	28
1.3.18. ucontext.h	29
1.3.19. unistd.h	30
1.3.20. utmp.h	30
1.3.21. utmpx.h	30
1.4. Interfaces for libm	31
1.4.1. Math	31
1.4.1.1. Interfaces for Math	31
1.5. Interfaces for libpthread	36
1.5.1. Realtime Threads	36
1.5.1.1. Interfaces for Realtime Threads	36
1.5.2. Advanced Realtime Threads	36
1.5.2.1. Interfaces for Advanced Realtime Threads	36
1.5.3. Posix Threads	36
1.5.3.1. Interfaces for Posix Threads	36
1.6. Interfaces for libgcc_s	39
1.6.1. Unwind Library	39
1.6.1.1. Interfaces for Unwind Library	39
1.7. Interface Definitions for libgcc_s	39
_Unwind_DeleteException	40
_Unwind_ForcedUnwind	41
_Unwind_GetGR	42
_Unwind_GetIP	42
_Unwind_GetLanguageSpecificData	42
_Unwind_GetRegionStart	43
_Unwind_RaiseException	44
_Unwind_Resume	45
_Unwind_SetGR	45
_Unwind_SetIP	45
1.8. Interfaces for libdl	45
1.8.1. Dynamic Loader	46
1.8.1.1. Interfaces for Dynamic Loader	46
1.9. Interfaces for libcrypt	46
1.9.1. Encryption	46
1.9.1.1. Interfaces for Encryption	46

II. Utility Libraries	48
2. Libraries	49
2.1. Interfaces for libz	49
2.1.1. Compression Library	49
2.1.1.1. Interfaces for Compression Library	49
2.2. Interfaces for libncurses	49
2.2.1. Curses	49
2.2.1.1. Interfaces for Curses	49
2.3. Interfaces for libutil	49
2.3.1. Utility Functions	50
2.3.1.1. Interfaces for Utility Functions	50
A. Alphabetical Listing of Interfaces	51
A.1. libgcc_s	51

List of Tables

1-1. libc Definition.....	1
1-2. libc - RPC Function Interfaces	1
1-3. libc - System Calls Function Interfaces	3
1-4. libc - Standard I/O Function Interfaces	5
1-5. libc - Standard I/O Data Interfaces	7
1-6. libc - Signal Handling Function Interfaces	7
1-7. libc - Signal Handling Data Interfaces.....	8
1-8. libc - Localization Functions Function Interfaces	8
1-9. libc - Localization Functions Data Interfaces	9
1-10. libc - Socket Interface Function Interfaces	9
1-11. libc - Socket Interface Deprecated Function Interfaces	10
1-12. libc - Wide Characters Function Interfaces	10
1-13. libc - String Functions Function Interfaces.....	12
1-14. libc - IPC Functions Function Interfaces	13
1-15. libc - Regular Expressions Function Interfaces	14
1-16. libc - Regular Expressions Deprecated Function Interfaces	14
1-17. libc - Regular Expressions Deprecated Data Interfaces.....	15
1-18. libc - Character Type Functions Function Interfaces.....	15
1-19. libc - Time Manipulation Function Interfaces	16
1-20. libc - Time Manipulation Deprecated Function Interfaces	16
1-21. libc - Time Manipulation Data Interfaces.....	17
1-22. libc - Terminal Interface Functions Function Interfaces.....	17
1-23. libc - System Database Interface Function Interfaces.....	18
1-24. libc - Language Support Function Interfaces.....	19
1-25. libc - Large File Support Function Interfaces	19
1-26. libc - Standard Library Function Interfaces	20
1-27. libc - Standard Library Data Interfaces	22
1-28. libm Definition	31
1-29. libm - Math Function Interfaces	31
1-30. libm - Math Data Interfaces.....	36
1-31. libpthread Definition	36
1-32. libpthread - Posix Threads Function Interfaces	37
1-33. libgcc_s Definition	39
1-34. libgcc_s - Unwind Library Function Interfaces	39
1-35. libdl Definition	46
1-36. libdl - Dynamic Loader Function Interfaces	46
1-37. libcrypt Definition	46
1-38. libcrypt - Encryption Function Interfaces	47
2-1. libz Definition.....	49
2-2. libncurses Definition	49
2-3. libutil Definition	49
2-4. libutil - Utility Functions Function Interfaces	50
A-1. libgcc_s Function Interfaces	51

I. Base Libraries

Chapter 1. Libraries

- 1 An LSB-conforming implementation shall support base libraries which provide interfaces for accessing the operating
 2 system, processor and other hardware in the system.
- 3 Only those interfaces that are unique to the Itanium™ platform are defined here. This section should be used in
 4 conjunction with the corresponding section in the Linux Standard Base Specification.

1.1. Program Interpreter/Dynamic Linker

- 5 The LSB specifies the Program Interpreter to be /lib/ld-lsb-ia64.so.2.

1.2. Interfaces for libc

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

7 **Table 1-1. libc Definition**

Library:	libc
SONAME:	libc.so.6.1

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

Large File Support

~~Linux Standard Base~~this specification

~~CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0, C606)~~SUSv2

~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

~~System V Interface Definition, SVID Issue 3 (ISBN 0201566524)~~

- 10 ~~System V Interface Definition, Fourth Edition SVID Issue 4~~

1.2.1. RPC

1.2.1.1. Interfaces for RPC

- 11 An LSB conforming implementation shall provide the architecture specific functions for RPC specified in Table 1-2,
 12 with the full functionality as described in the referenced underlying specification.

14 **Table 1-2. libc - RPC Function Interfaces**

authnone_create(GLIBC_2.2) authnone_create(GLIBC_2.2) [1]	pmap_unset(GLIBC_2.2) pmap_unset(GLIBC_2.2) [2]	svcerr_weakauth(GLIBC_2.2) svcerr_weakauth(GLIBC_2.2) [3]	xdr_float(GLIBC_2.2) xdr_float(GLIBC_2.2) [3]	xdr_u_char(GLIBC_2.2) xdr_u_char(GLIBC_2.2) [3]
clnt_create(GLIBC_2.2) clnt_create(GLIBC_2.2)	setdomainname(GLIBC_2.2) setdomainname	svctcp_create(GLIBC_2.2) svctcp_create	xdr_free(GLIBC_2.2) xdr_free(GLIBC_2.2)	xdr_u_int(GLIBC_2.2) xdr_u_int(GLIBC_2.2)

BC_2.2) [1]	name(GLIBC_2.2) [2]	(GLIBC_2.2) [2]	2.2) [3]	C_2.2) [2]
clnt_pcreateerror(GLIBC_2.2)clnt_pcreateerror(GLIBC_2.2) [1]	svc_getreqset(GLIBC_2.2)svc_getreqset(GLIBC_2.2) [3]	sveudp_create(GLIBC_2.2)svcudp_create(GLIBC_2.2) [2]	xdr_int(GLIBC_2.2)xdr_int(GLIBC_2.2) [3]	xdr_u_long(GLIBC_2.2)xdr_u_long(GLIBC_2.2) [3]
clnt_perrno(GLIBC_2.2)clnt_perrno(GLIBC_2.2) [1]	svc_register(GLIBC_2.2)svc_register(GLIBC_2.2) [2]	xdr_accepted_reply(GLIBC_2.2)xdr_accepted_reply(GLIBC_2.2) [3]	xdr_long(GLIBC_2.2)xdr_long(GLIBC_2.2) [3]	xdr_u_short(GLIBC_2.2)xdr_u_short(GLIBC_2.2) [3]
clnt_perror(GLIBC_2.2)clnt_perror(GLIBC_2.2) [1]	svc_run(GLIBC_2.2)svc_run(GLIBC_2.2) [2]	xdr_array(GLIBC_2.2)xdr_array(GLIBC_2.2) [3]	xdr_opaque(GLIBC_2.2)xdr_opaque(GLIBC_2.2) [3]	xdr_union(GLIBC_2.2)xdr_union(GLIBC_2.2) [3]
clnt_screateerror(GLIBC_2.2)clnt_screateerror(GLIBC_2.2) [1]	svc_sendreply(GLIBC_2.2)svc_sendreply(GLIBC_2.2) [2]	xdr_bool(GLIBC_2.2)xdr_bool(GLIBC_2.2) [3]	xdr_opaque_auth(GLIBC_2.2)xdr_opaque_auth(GLIBC_2.2) [3]	xdr_vector(GLIBC_2.2)xdr_vector(GLIBC_2.2) [3]
clnt_serrno(GLIBC_2.2)clnt_serrno(GLIBC_2.2) [1]	svcerr_auth(GLIBC_2.2)svcerr_auth(GLIBC_2.2) [3]	xdr_bytes(GLIBC_2.2)xdr_bytes(GLIBC_2.2) [3]	xdr_pointer(GLIBC_2.2)xdr_pointer(GLIBC_2.2) [3]	xdr_void(GLIBC_2.2)xdr_void(GLIBC_2.2) [3]
clnt_serror(GLIBC_2.2)clnt_serror(GLIBC_2.2) [1]	svcerr_decode(GLIBC_2.2)svcerr_decode(GLIBC_2.2) [3]	xdr_callhdr(GLIBC_2.2)xdr_callhdr(GLIBC_2.2) [3]	xdr_reference(GLIBC_2.2)xdr_reference(GLIBC_2.2) [3]	xdr_wrapstring(GLIBC_2.2)xdr_wrapstring(GLIBC_2.2) [3]
getdomainname(GLIBC_2.2)getdomainname(GLIBC_2.2) [2]	svcerr_noproc(GLIBC_2.2)svcerr_noproc(GLIBC_2.2) [3]	xdr_callmsg(GLIBC_2.2)xdr_callmsg(GLIBC_2.2) [3]	xdr_rejected_reply(GLIBC_2.2)xdr_rejected_reply(GLIBC_2.2) [3]	xdrmem_create(GLIBC_2.2)xdrmem_create(GLIBC_2.2) [3]
key_decryptsession(GLIBC_2.2)key_decryptsession(GLIBC_2.2) [3]	svcerr_noprog(GLIBC_2.2)svcerr_noprog(GLIBC_2.2) [3]	xdr_char(GLIBC_2.2)xdr_char(GLIBC_2.2) [3]	xdr_replymsg(GLIBC_2.2)xdr_replymsg(GLIBC_2.2) [3]	xdrrec_create(GLIBC_2.2)xdrrec_create(GLIBC_2.2) [3]
pmap_getport(GLIBC_2.2)pmap_getport(GLIBC_2.2) [2]	svcerr_progvers(GLIBC_2.2)svcerr_progvers(GLIBC_2.2) [3]	xdr_double(GLIBC_2.2)xdr_double(GLIBC_2.2) [3]	xdr_short(GLIBC_2.2)xdr_short(GLIBC_2.2) [3]	xdrrec_eof(GLIBC_2.2)xdrrec_eof(GLIBC_2.2) [3]
pmap_set(GLIBC_2.2)pmap_set(GLIBC_2.2) [2]	svcerr_systemerr(GLIBC_2.2)svcerr_systemerr(GLIBC_2.2) [3]	xdr_enum(GLIBC_2.2)xdr_enum(GLIBC_2.2) [3]	xdr_string(GLIBC_2.2)xdr_string(GLIBC_2.2) [3]	

15

16 *Referenced Specification(s)*

17 [1]. System V Interface Definition, Fourth Edition SVID Issue 4

- 18 [2]. Linux Standard Base this specification
 19 [3]. System V Interface Definition, SVID Issue 3 (ISBN 0201566524)

1.2.2. System Calls

1.2.2.1. Interfaces for System Calls

21 An LSB conforming implementation shall provide the architecture specific functions for System Calls specified in
 22 Table 1-3, with the full functionality as described in the referenced underlying specification.

23 **Table 1-3. libc - System Calls Function Interfaces**

<code>__fxstat(GLIBC_2.2)</code> <code>__fxstat(GLIBC_2.2)</code> [1]	<code>fchmod(GLIBC_2.2)</code> <code>fchmod(GLIBC_2.2)</code> [2]	<code>getwd(GLIBC_2.2)</code> <code>getwd(GLIBC_2.2)</code> [2]	<code>read(GLIBC_2.2)</code> <code>read(GLIBC_2.2)</code> [2]	<code>setrlimit(GLIBC_2.2)</code> <code>setrlimit(GLIBC_2.2)</code> [2]
<code>__getpgid(GLIBC_2.2)</code> <code>__getpgid(GLIBC_2.2)</code> [1]	<code>fchown(GLIBC_2.2)</code> <code>fchown(GLIBC_2.2)</code> [2]	<code>initgroups(GLIBC_2.2)</code> <code>initgroups(GLIBC_2.2)</code> [1]	<code>readdir(GLIBC_2.2)</code> <code>readdir(GLIBC_2.2)</code> [2]	<code>setrlimit64(GLIBC_2.2)</code> <code>setrlimit64(GLIBC_2.2)</code> [3]
<code>__lxstat(GLIBC_2.2)</code> <code>__lxstat(GLIBC_2.2)</code> [1]	<code>fentl(GLIBC_2.2)</code> <code>fentl(GLIBC_2.2)</code> [1]	<code>ioctl(GLIBC_2.2)</code> <code>ioctl(GLIBC_2.2)</code> [1]	<code>readdir_r(GLIBC_2.2)</code> <code>readdir_r(GLIBC_2.2)</code> [2]	<code>setsid(GLIBC_2.2)</code> <code>setsid(GLIBC_2.2)</code> [2]
<code>__xmknod(GLIBC_2.2)</code> <code>__xmknod(GLIBC_2.2)</code> [1]	<code>fdatasync(GLIBC_2.2)</code> <code>fdatasync(GLIBC_2.2)</code> [2]	<code>kill(GLIBC_2.2)</code> <code>kill(GLIBC_2.2)</code> [1]	<code>readlink(GLIBC_2.2)</code> <code>readlink(GLIBC_2.2)</code> [2]	<code>setuid(GLIBC_2.2)</code> <code>setuid(GLIBC_2.2)</code> [2]
<code>__xstat(GLIBC_2.2)</code> <code>__xstat(GLIBC_2.2)</code> [1]	<code>flock(GLIBC_2.2)</code> <code>flock(GLIBC_2.2)</code> [1]	<code>killpg(GLIBC_2.2)</code> <code>killpg(GLIBC_2.2)</code> [2]	<code>readv(GLIBC_2.2)</code> <code>readv(GLIBC_2.2)</code> [2]	<code>sleep(GLIBC_2.2)</code> <code>sleep(GLIBC_2.2)</code> [2]
<code>access(GLIBC_2.2)</code> <code>access(GLIBC_2.2)</code> [2]	<code>fork(GLIBC_2.2)</code> <code>fork(GLIBC_2.2)</code> [2]	<code>lchown(GLIBC_2.2)</code> <code>lchown(GLIBC_2.2)</code> [2]	<code>rename(GLIBC_2.2)</code> <code>rename(GLIBC_2.2)</code> [2]	<code>statvfs(GLIBC_2.2)</code> <code>statvfs(GLIBC_2.2)</code> [2]
<code>aect(GLIBC_2.2)</code> <code>aect(GLIBC_2.2)</code> [1]	<code>fstatvfs(GLIBC_2.2)</code> <code>fstatvfs(GLIBC_2.2)</code> [2]	<code>link(GLIBC_2.2)</code> <code>link(GLIBC_2.2)</code> [2]	<code>rmdir(GLIBC_2.2)</code> <code>rmdir(GLIBC_2.2)</code> [2]	<code>stime(GLIBC_2.2)</code> <code>stime(GLIBC_2.2)</code> [1]
<code>alarm(GLIBC_2.2)</code> <code>alarm(GLIBC_2.2)</code> [2]	<code>fsync(GLIBC_2.2)</code> <code>fsync(GLIBC_2.2)</code> [2]	<code>lockf(GLIBC_2.2)</code> <code>lockf(GLIBC_2.2)</code> [2]	<code>sbrk(GLIBC_2.2)</code> <code>sbrk(GLIBC_2.2)</code> [4]	<code>symlink(GLIBC_2.2)</code> <code>symlink(GLIBC_2.2)</code> [2]
<code>brk(GLIBC_2.2)</code> <code>brk(GLIBC_2.2)</code> [4]	<code>ftime(GLIBC_2.2)</code> <code>ftime(GLIBC_2.2)</code> [2]	<code>lseek(GLIBC_2.2)</code> <code>lseek(GLIBC_2.2)</code> [2]	<code>sched_get_priority_max(GLIBC_2.2)</code> <code>sched_get_priority_max(GLIBC_2.2)</code> [2]	<code>sync(GLIBC_2.2)</code> <code>sync(GLIBC_2.2)</code> [2]
<code>chdir(GLIBC_2.2)</code> <code>chdir(GLIBC_2.2)</code> [2]	<code>ftruncate(GLIBC_2.2)</code> <code>ftruncate(GLIBC_2.2)</code> [2]	<code>mkdir(GLIBC_2.2)</code> <code>mkdir(GLIBC_2.2)</code> [2]	<code>sched_get_priority_min(GLIBC_2.2)</code> <code>sched_get_priority_min(GLIBC_2.2)</code> [2]	<code>sysconf(GLIBC_2.2)</code> <code>sysconf(GLIBC_2.2)</code> [2]

			n(GLIBC_2.2) [2]	
chmod(GLIBC_2.2) chmod(GLIBC_2.2) [2]	getecontext(GLIBC_2.2) getecontext(GLIBC_2.2) [2]	mkfifo(GLIBC_2.2) mkfifo(GLIBC_2.2) [2]	sched_getparam(GLIBC_2.2) sched_getparam(GLIBC_2.2) [2]	time(GLIBC_2.2) time(GLIBC_2.2) [2]
chown(GLIBC_2.2) chown(GLIBC_2.2) [2]	getegid(GLIBC_2.2) getegid(GLIBC_2.2) [2]	mlock(GLIBC_2.2) mlock(GLIBC_2.2) [2]	sched_getscheduler(GLIBC_2.2) sched_getscheduler(GLIBC_2.2) [2]	times(GLIBC_2.2) times(GLIBC_2.2) [2]
chroot(GLIBC_2.2) chroot(GLIBC_2.2) [4]	geteuid(GLIBC_2.2) geteuid(GLIBC_2.2) [2]	mlockall(GLIBC_2.2) mlockall(GLIBC_2.2) [2]	sched_rr_get_interval(GLIBC_2.2) sched_rr_get_interval(GLIBC_2.2) [2]	truncate(GLIBC_2.2) truncate(GLIBC_2.2) [2]
clock(GLIBC_2.2) clock(GLIBC_2.2) [2]	getgid(GLIBC_2.2) getgid(GLIBC_2.2) [2]	mmap(GLIBC_2.2) mmap(GLIBC_2.2) [2]	sched_setparam(GLIBC_2.2) sched_setparam(GLIBC_2.2) [2]	ulimit(GLIBC_2.2) ulimit(GLIBC_2.2) [2]
close(GLIBC_2.2) close(GLIBC_2.2) [2]	getgroups(GLIBC_2.2) getgroups(GLIBC_2.2) [2]	mprotect(GLIBC_2.2) mprotect(GLIBC_2.2) [2]	sched_setscheduler(GLIBC_2.2) sched_setscheduler(GLIBC_2.2) [2]	umask(GLIBC_2.2) umask(GLIBC_2.2) [2]
closedir(GLIBC_2.2) closedir(GLIBC_2.2) [2]	getitimer(GLIBC_2.2) getitimer(GLIBC_2.2) [2]	msync(GLIBC_2.2) msync(GLIBC_2.2) [2]	sched_yield(GLIBC_2.2) sched_yield(GLIBC_2.2) [2]	uname(GLIBC_2.2) uname(GLIBC_2.2) [2]
creat(GLIBC_2.2) creat(GLIBC_2.2) [1]	getloadavg(GLIBC_2.2) getloadavg(GLIBC_2.2) [1]	munlock(GLIBC_2.2) munlock(GLIBC_2.2) [2]	select(GLIBC_2.2) select(GLIBC_2.2) [2]	unlink(GLIBC_2.2) unlink(GLIBC_2.2) [1]
dup(GLIBC_2.2) dup(GLIBC_2.2) [2]	getpagesize(GLIBC_2.2) getpagesize(GLIBC_2.2) [4]	munlockall(GLIBC_2.2) munlockall(GLIBC_2.2) [2]	setecontext(GLIBC_2.2) setcontext(GLIBC_2.2) [2]	utime(GLIBC_2.2) utime(GLIBC_2.2) [2]
dup2(GLIBC_2.2) dup2(GLIBC_2.2) [2]	getpgid(GLIBC_2.2) getpgid(GLIBC_2.2) [2]	munmap(GLIBC_2.2) munmap(GLIBC_2.2) [2]	setegid(GLIBC_2.2) setegid(GLIBC_2.2) [2]	utimes(GLIBC_2.2) utimes(GLIBC_2.2) [2]
execl(GLIBC_2.2) execl(GLIBC_2.2) [2]	getpgrp(GLIBC_2.2) getpgrp(GLIBC_2.2) [2]	nanosleep(GLIBC_2.2) nanosleep(GLIBC_2.2) [2]	seteuid(GLIBC_2.2) seteuid(GLIBC_2.2) [2]	vfork(GLIBC_2.2) vfork(GLIBC_2.2) [2]
execle(GLIBC_2.2) execle(GLIBC_2.2) [2]	getpid(GLIBC_2.2) getpid(GLIBC_2.2) [2]	nice(GLIBC_2.2) nice(GLIBC_2.2) [2]	setgid(GLIBC_2.2) setgid(GLIBC_2.2) [2]	wait(GLIBC_2.2) wait(GLIBC_2.2) [2]
execlp(GLIBC_2.2) execlp(GLIBC_2.2)	getppid(GLIBC_2.2) getppid(GLIBC_2.2)	open(GLIBC_2.2) open(GLIBC_2.2) [1]	setitimer(GLIBC_2.2) setitimer(GLIBC_2.2)	wait3(GLIBC_2.2) wait3(GLIBC_2.2)

[2]	2) [2]		2.2) [2]	[1]
<code>execv(GLIBC_2.2)</code> <code>execv(GLIBC_2.2)</code> [2]	<code>getpriority(GLIBC_2.2)</code> <code>getpriority(GLIBC_2.2)</code> [2]	<code>opendir(GLIBC_2.2)</code> <code>opendir(GLIBC_2.2)</code> [2]	<code>setpgid(GLIBC_2.2)</code> <code>setpgid(GLIBC_2.2)</code> [2]	<code>wait4(GLIBC_2.2)</code> <code>wait4(GLIBC_2.2)</code> [1]
<code>execve(GLIBC_2.2)</code> <code>execve(GLIBC_2.2)</code> [2]	<code>getrlimit(GLIBC_2.2)</code> <code>getrlimit(GLIBC_2.2)</code> [2]	<code>pathconf(GLIBC_2.2)</code> <code>pathconf(GLIBC_2.2)</code> [2]	<code>setpgrp(GLIBC_2.2)</code> <code>setpgrp(GLIBC_2.2)</code> [2]	<code>waitpid(GLIBC_2.2)</code> <code>waitpid(GLIBC_2.2)</code> [1]
<code>execvp(GLIBC_2.2)</code> <code>execvp(GLIBC_2.2)</code> [2]	<code>getrusage(GLIBC_2.2)</code> <code>getrusage(GLIBC_2.2)</code> [2]	<code>pause(GLIBC_2.2)</code> <code>pause(GLIBC_2.2)</code> [2]	<code>setpriority(GLIBC_2.2)</code> <code>setpriority(GLIBC_2.2)</code> [2]	<code>write(GLIBC_2.2)</code> <code>write(GLIBC_2.2)</code> [2]
<code>exit(GLIBC_2.2)</code> <code>exit(GLIBC_2.2)</code> [2]	<code>getsid(GLIBC_2.2)</code> <code>getsid(GLIBC_2.2)</code> [2]	<code>pipe(GLIBC_2.2)</code> <code>pipe(GLIBC_2.2)</code> [2]	<code>setregid(GLIBC_2.2)</code> <code>setregid(GLIBC_2.2)</code> [2]	<code>writew(GLIBC_2.2)</code> <code>writew(GLIBC_2.2)</code> [2]
<code>fcntl(GLIBC_2.2)</code> <code>fcntl(GLIBC_2.2)</code> [2]	<code>getuid(GLIBC_2.2)</code> <code>getuid(GLIBC_2.2)</code> [2]	<code>poll(GLIBC_2.2)</code> <code>poll(GLIBC_2.2)</code> [2]	<code>setreuid(GLIBC_2.2)</code> <code>setreuid(GLIBC_2.2)</code> [2]	

24

25 *Referenced Specification(s)*

26 [1]. Linux Standard Base this specification

27 [2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
28 V3)

29 [3]. Large File Support

30 [4]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0,
31 €606) SUSv2

1.2.3. Standard I/O

32 1.2.3.1. Interfaces for Standard I/O

33 An LSB conforming implementation shall provide the architecture specific functions for Standard I/O specified in
34 Table 1-4, with the full functionality as described in the referenced underlying specification.35 **Table 1-4. libc - Standard I/O Function Interfaces**

<code>_IO_feof(GLIBC_2.2)</code> <code>_IO_feof(GLIBC_2.2)</code> [1]	<code>fgetpos(GLIBC_2.2)</code> <code>fgetpos(GLIBC_2.2)</code> [2]	<code>fsetpos(GLIBC_2.2)</code> <code>fsetpos(GLIBC_2.2)</code> [2]	<code>putchar(GLIBC_2.2)</code> <code>putchar(GLIBC_2.2)</code> [2]	<code>sscanf(GLIBC_2.2)</code> <code>sscanf(GLIBC_2.2)</code> [2]
<code>_IO_getc(GLIBC_2.2)</code> <code>_IO_getc(GLIBC_2.2)</code> [1]	<code>fgets(GLIBC_2.2)</code> <code>fgets(GLIBC_2.2)</code> [2]	<code>ftell(GLIBC_2.2)</code> <code>ftell(GLIBC_2.2)</code> [2]	<code>putchar_unlocked(GLIBC_2.2)</code> <code>putchar_unlocked(GLIBC_2.2)</code> [2]	<code>telldir(GLIBC_2.2)</code> <code>telldir(GLIBC_2.2)</code> [2]
<code>_IO_putc(GLIBC_2.2)</code> <code>_IO_putc(GLIBC_2.2)</code> [1]	<code>fgetwc_unlocked(GLIBC_2.2)</code> <code>fgetwc_unlocked(GLIBC_2.2)</code> [2]	<code>ftello(GLIBC_2.2)</code> <code>ftello(GLIBC_2.2)</code> [2]	<code>puts(GLIBC_2.2)</code> <code>puts(GLIBC_2.2)</code> [2]	<code>tempnam(GLIBC_2.2)</code> <code>tempnam(GLIBC_2.2)</code> [2]

_2.2) [1]	nlocked(GLIBC_2.2) [1]	[2]		_2.2) [2]
_IO_puts(GLIBC_2.2)_IO_puts(GLIBC_2.2) [1]	fileno(GLIBC_2.2)fileno(GLIBC_2.2) [2]	fwrite(GLIBC_2.2)fwrite(GLIBC_2.2) [2]	putw(GLIBC_2.2)putw(GLIBC_2.2) [3]	ungetc(GLIBC_2.2)ungetc(GLIBC_2.2) [2]
asprintf(GLIBC_2.2)asprintf(GLIBC_2.2) [1]	flockfile(GLIBC_2.2)flockfile(GLIBC_2.2) [2]	getc(GLIBC_2.2)getc(GLIBC_2.2) [2]	remove(GLIBC_2.2)remove(GLIBC_2.2) [2]	vasprintf(GLIBC_2.2)vasprintf(GLIBC_2.2) [1]
clearerr(GLIBC_2.2)clearerr(GLIBC_2.2) [2]	fopen(GLIBC_2.2)fopen(GLIBC_2.2) [1]	getc_unlocked(GLIBC_2.2)getc_unlocked(GLIBC_2.2) [2]	rewind(GLIBC_2.2)rewind(GLIBC_2.2) [2]	vdprintf(GLIBC_2.2)vdprintf(GLIBC_2.2) [1]
etermid(GLIBC_2.2)etermid(GLIBC_2.2) [2]	fprintf(GLIBC_2.2)fprintf(GLIBC_2.2) [2]	getchar(GLIBC_2.2)getchar(GLIBC_2.2) [2]	rewinddir(GLIBC_2.2)rewinddir(GLIBC_2.2) [2]	vfprintf(GLIBC_2.2)vfprintf(GLIBC_2.2) [2]
fclose(GLIBC_2.2)fclose(GLIBC_2.2) [2]	fputc(GLIBC_2.2)fputc(GLIBC_2.2) [2]	getchar_unlocked(GLIBC_2.2)getchar_unlocked(GLIBC_2.2) [2]	scanf(GLIBC_2.2)scanf(GLIBC_2.2) [2]	vprintf(GLIBC_2.2)vprintf(GLIBC_2.2) [2]
fdopen(GLIBC_2.2)fdopen(GLIBC_2.2) [2]	fputs(GLIBC_2.2)fputs(GLIBC_2.2) [2]	getw(GLIBC_2.2)getw(GLIBC_2.2) [3]	seekdir(GLIBC_2.2)seekdir(GLIBC_2.2) [2]	vsprintf(GLIBC_2.2)vsprintf(GLIBC_2.2) [2]
feof(GLIBC_2.2)feof(GLIBC_2.2) [2]	fread(GLIBC_2.2)fread(GLIBC_2.2) [2]	pclose(GLIBC_2.2)pclose(GLIBC_2.2) [2]	setbuf(GLIBC_2.2)setbuf(GLIBC_2.2) [2]	vsprintf(GLIBC_2.2)vsprintf(GLIBC_2.2) [2]
ferror(GLIBC_2.2)ferror(GLIBC_2.2) [2]	freopen(GLIBC_2.2)freopen(GLIBC_2.2) [1]	popen(GLIBC_2.2)popen(GLIBC_2.2) [2]	setbuffer(GLIBC_2.2)setbuffer(GLIBC_2.2) [1]	
fflush(GLIBC_2.2)fflush(GLIBC_2.2) [2]	fscanf(GLIBC_2.2)fscanf(GLIBC_2.2) [2]	printf(GLIBC_2.2)printf(GLIBC_2.2) [2]	setvbuf(GLIBC_2.2)setvbuf(GLIBC_2.2) [2]	
fflush_unlocked(GLIBC_2.2)fflush_unlocked(GLIBC_2.2) [1]	fseek(GLIBC_2.2)fseek(GLIBC_2.2) [2]	putc(GLIBC_2.2)putc(GLIBC_2.2) [2]	snprintf(GLIBC_2.2)snprintf(GLIBC_2.2) [2]	
fgetc(GLIBC_2.2)fgetc(GLIBC_2.2) [2]	fseeko(GLIBC_2.2)fseeko(GLIBC_2.2) [2]	putc_unlocked(GLIBC_2.2)putc_unlocked(GLIBC_2.2) [2]	sprintf(GLIBC_2.2)sprintf(GLIBC_2.2) [2]	

36

37 *Referenced Specification(s)*38 [1]. ~~Linux Standard Base~~this specification

39 [2]. ISO/IEC 9945: POSIX (2003-Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
40 V3)

41 [3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0,
42 €606) SUSv2

43 An LSB conforming implementation shall provide the architecture specific data interfaces for Standard I/O specified
44 in Table 1-5, with the full functionality as described in the referenced underlying specification.

45 **Table 1-5. libc - Standard I/O Data Interfaces**

stderr(GLIBC_2.2)s tderr(GLIBC_2.2) [1]	stdin(GLIBC_2.2)st din(GLIBC_2.2) [1]	stdout(GLIBC_2.2)s tdout(GLIBC_2.2) [1]		
-----------------------------------------------	------------------------------------------	-----------------------------------------------	--	--

47 *Referenced Specification(s)*

48 [1]. ISO/IEC 9945: POSIX (2003-Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
49 V3)

1.2.4. Signal Handling

1.2.4.1. Interfaces for Signal Handling

51 An LSB conforming implementation shall provide the architecture specific functions for Signal Handling specified in
52 Table 1-6, with the full functionality as described in the referenced underlying specification.

53 **Table 1-6. libc - Signal Handling Function Interfaces**

__libc_current_sigrt max(GLIBC_2.2) __ libc_current_sigrtm ax(GLIBC_2.2) [1]	sigaddset(GLIBC_2 _2) sigaddset(GLIBC _2.2) [2]	sighold(GLIBC_2.2) sighold(GLIBC_2. 2) [2]	sigpause(GLIBC_2. 2) sigpause(GLIBC_ 2.2) [2]	sigsuspend(GLIBC_ 2.2) sigsuspend(GLI BC_2.2) [2]
__libc_current_sigrt min(GLIBC_2.2) __l ibc_current_sigrtmi n(GLIBC_2.2) [1]	sigaltstack(GLIBC_ 2.2) sigaltstack(GLI BC_2.2) [2]	sigignore(GLIBC_2 _2) sigignore(GLIBC _2.2) [2]	sigpending(GLIBC_ 2.2) sigpending(GLI BC_2.2) [2]	sigtimedwait(GLIB C_2.2) sigtimedwait(GLIBC_2.2) [2]
__sigsetjmp(GLIBC _2.2) __sigsetjmp(G LIBC_2.2) [1]	sigandset(GLIBC_2 _2) sigandset(GLIBC _2.2) [1]	siginterrupt(GLIBC _2.2) siginterrupt(G LIBC_2.2) [2]	sigprocmask(GLIB C_2.2) sigprocmask(GLIBC_2.2) [2]	sigwait(GLIBC_2.2) sigwait(GLIBC_2. 2) [2]
__sysv_signal(GLI BC_2.2) __sysv_sig nal(GLIBC_2.2) [1]	sigblock(GLIBC_2. 2) sigblock(GLIBC_ 2.2) [1]	sigisemptyset(GLIB C_2.2) sigisemptyset (GLIBC_2.2) [1]	sigqueue(GLIBC_2. 2) sigqueue(GLIBC_ 2.2) [2]	sigwaitinfo(GLIBC _2.2) sigwaitinfo(GL IBC_2.2) [2]
bsd_signal(GLIBC_ 2.2) bsd_signal(GLI BC_2.2) [2]	sigdelset(GLIBC_2. 2) sigdelset(GLIBC_ 2.2) [2]	sigismember(GLIB C_2.2) sigismember(GLIBC_2.2) [2]	sigrelse(GLIBC_2.2) sigrelse(GLIBC_2. 2) [2]	
psignal(GLIBC_2.2) psignal(GLIBC_2.	sigemptyset(GLIBC _2.2) sigemptyset(G	siglongjmp(GLIBC _2.2) siglongjmp(GL	sigreturn(GLIBC_2. 2) sigreturn(GLIBC_	

2) [1]	LIBC_2.2) [2]	IBC_2.2) [2]	2.2) [1]	
raise(GLIBC_2.2) raise(GLIBC_2.2) [2]	sigfillset(GLIBC_2.2) sigfillset(GLIBC_2.2) [2]	signal(GLIBC_2.2) signal(GLIBC_2.2) [2]	sigset(GLIBC_2.2) sigset(GLIBC_2.2) [2]	
sigaction(GLIBC_2.2) sigaction(GLIBC_2.2) [2]	siggetmask(GLIBC_2.2) siggetmask(GLIBC_2.2) [1]	sigorset(GLIBC_2.2) sigorset(GLIBC_2.2) [1]	sigstack(GLIBC_2.2) sigstack(GLIBC_2.2) [3]	

54

55 *Referenced Specification(s)*56 [1]. ~~Linux Standard Base~~this specification57 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)~~58 ~~V3)~~59 [3]. ~~CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1 85912 181 0,~~60 ~~€606)SUSv2~~

61 An LSB conforming implementation shall provide the architecture specific data interfaces for Signal Handling

62 specified in Table 1-7, with the full functionality as described in the referenced underlying specification.

63 **Table 1-7. libc - Signal Handling Data Interfaces**

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

64

65 *Referenced Specification(s)*66 [1]. ~~Linux Standard Base~~this specification

1.2.5. Localization Functions

1.2.5.1. Interfaces for Localization Functions

68 An LSB conforming implementation shall provide the architecture specific functions for Localization Functions

69 specified in Table 1-8, with the full functionality as described in the referenced underlying specification.

70 **Table 1-8. libc - Localization Functions Function Interfaces**

bind_textdomain_codeset(GLIBC_2.2) bind_textdomain_codeset(GLIBC_2.2) [1]	catopen(GLIBC_2.2) catopen(GLIBC_2.2) [2]	dngettext(GLIBC_2.2) dngettext(GLIBC_2.2) [1]	iconv_open(GLIBC_2.2) iconv_open(GLIBC_2.2) [2]	setlocale(GLIBC_2.2) setlocale(GLIBC_2.2) [2]
bindtextdomain(GLIBC_2.2) bindtextdomain(GLIBC_2.2) [1]	dcgettext(GLIBC_2.2) dcgettext(GLIBC_2.2) [1]	gettext(GLIBC_2.2) gettext(GLIBC_2.2) [1]	localeconv(GLIBC_2.2) localeconv(GLIBC_2.2) [2]	textdomain(GLIBC_2.2) textdomain(GLIBC_2.2) [1]

<code>eatclose(GLIBC_2.2)</code> <code>catclose(GLIBC_2.2)</code> [2]	<code>dengettext(GLIBC_2.2)</code> <code>dcngettext(GLIBC_2.2)</code> [1]	<code>ieconv(GLIBC_2.2)</code> <code>iconv(GLIBC_2.2)</code> [2]	<code>ngettext(GLIBC_2.2)</code> <code>ncngettext(GLIBC_2.2)</code> [1]	
<code>eatgets(GLIBC_2.2)</code> <code>catgets(GLIBC_2.2)</code> [2]	<code>dgettext(GLIBC_2.2)</code> <code>dcgettext(GLIBC_2.2)</code> [1]	<code>ieconv_close(GLIBC_2.2)</code> <code>iconv_close(GLIBC_2.2)</code> [2]	<code>nl_langinfo(GLIBC_2.2)</code> <code>ncn_langinfo(GLIBC_2.2)</code> [2]	

71

72 *Referenced Specification(s)*73 [1]. ~~Linux Standard Base~~this specification74 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)~~75 ~~⋮~~)76 An LSB conforming implementation shall provide the architecture specific data interfaces for Localization Functions
77 specified in Table 1-9, with the full functionality as described in the referenced underlying specification.78 **Table 1-9. libc - Localization Functions Data Interfaces**

<code>_nl_msg_cat_cntr(GLIBC_2.2)</code> [1]				
----------------------------------------------	--	--	--	--

79

80 *Referenced Specification(s)*81 [1]. ~~Linux Standard Base~~this specification

1.2.6. Socket Interface

1.2.6.1. Interfaces for Socket Interface

83 An LSB conforming implementation shall provide the architecture specific functions for Socket Interface specified in
84 Table 1-10, with the full functionality as described in the referenced underlying specification.85 **Table 1-10. libc - Socket Interface Function Interfaces**

<code>__h_errno_location(GLIBC_2.2)</code> <code>__h_errno_location(GLIBC_2.2)</code> [1]	<code>gethostid(GLIBC_2.2)</code> <code>gethostid(GLIBC_2.2)</code> [2]	<code>listen(GLIBC_2.2)</code> <code>listen(GLIBC_2.2)</code> [2]	<code>sendmsg(GLIBC_2.2)</code> <code>sendmsg(GLIBC_2.2)</code> [2]	<code>socketpair(GLIBC_2.2)</code> <code>socketpair(GLIBC_2.2)</code> [2]
<code>accept(GLIBC_2.2)</code> <code>accept(GLIBC_2.2)</code> [2]	<code>gethostname(GLIBC_2.2)</code> <code>gethostname(GLIBC_2.2)</code> [2]	<code>recv(GLIBC_2.2)</code> <code>recv(GLIBC_2.2)</code> [2]	<code>sendto(GLIBC_2.2)</code> <code>sendto(GLIBC_2.2)</code> [2]	
<code>bind(GLIBC_2.2)</code> <code>bind(GLIBC_2.2)</code> [2]	<code>getpeername(GLIBC_2.2)</code> <code>getpeername(GLIBC_2.2)</code> [2]	<code>recvfrom(GLIBC_2.2)</code> <code>recvfrom(GLIBC_2.2)</code> [2]	<code>setsockopt(GLIBC_2.2)</code> <code>setsockopt(GLIBC_2.2)</code> [1]	
<code>bindresvport(GLIBC_2.2)</code> <code>bindresvport(GLIBC_2.2)</code>	<code>getsockname(GLIBC_2.2)</code> <code>getsockname(GLIBC_2.2)</code>	<code>recvmsg(GLIBC_2.2)</code> <code>recvmsg(GLIBC_2.2)</code>	<code>shutdown(GLIBC_2.2)</code> <code>shutdown(GLIBC_2.2)</code>	

GLIBC_2.2) [1]	(GLIBC_2.2) [2]	2.2) [2]	C_2.2) [2]	
econnect(GLIBC_2.2) connect(GLIBC_2.2) [2]	getsockopt(GLIBC_2.2) getsockopt(GLIBC_2.2) [2]	send(GLIBC_2.2) send(GLIBC_2.2) [2]	socket(GLIBC_2.2) socket(GLIBC_2.2) [2]	

86

87 *Referenced Specification(s)*88 [1]. ~~Linux Standard Base~~this specification89 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

90

91 An LSB conforming implementation shall provide the architecture specific deprecated functions for Socket Interface
92 specified in Table 1-11, with the full functionality as described in the referenced underlying specification.93 These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn
94 in future releases of this specification.95 **Table 1-11. libc - Socket Interface Deprecated Function Interfaces**

gethostbyname_r(GLIBC_2.2) gethostbyname_r(GLIBC_2.2) [1]				
------------------------------------------------------------------------------------	--	--	--	--

96

97 *Referenced Specification(s)*98 [1]. ~~Linux Standard Base~~this specification

1.2.7. Wide Characters

1.2.7.1. Interfaces for Wide Characters

99
100 An LSB conforming implementation shall provide the architecture specific functions for Wide Characters specified in
101 Table 1-12, with the full functionality as described in the referenced underlying specification.102 **Table 1-12. libc - Wide Characters Function Interfaces**

__westod_internal(GLIBC_2.2) __westod_internal(GLIBC_2.2) [1]	mbsinit(GLIBC_2.2) mbsinit(GLIBC_2.2) [2]	vwscanf(GLIBC_2.2) vwscanf(GLIBC_2.2) [2]	wesnlen(GLIBC_2.2) wcsnlen(GLIBC_2.2) [1]	westoumax(GLIBC_2.2) wcstoumax(GLIBC_2.2) [2]
__westof_internal(GLIBC_2.2) __westof_internal(GLIBC_2.2) [1]	mbsnrtowcs(GLIBC_2.2) mbsnrtowcs(GLIBC_2.2) [1]	wepcpy(GLIBC_2.2) wcpcpy(GLIBC_2.2) [1]	wesnrtombs(GLIBC_2.2) wcsnrtombs(GLIBC_2.2) [1]	westouq(GLIBC_2.2) wcstouq(GLIBC_2.2) [1]
__westol_internal(GLIBC_2.2) __westol_internal(GLIBC_2.2) [1]	mbsrtowcs(GLIBC_2.2) mbsrtowcs(GLIBC_2.2) [2]	wepncpy(GLIBC_2.2) wcpcpy(GLIBC_2.2) [1]	wesprk(GLIBC_2.2) wcspbrk(GLIBC_2.2) [2]	weswcs(GLIBC_2.2) wcswcs(GLIBC_2.2) [2]

<code>__westold_internal(GLIBC_2.2) __wcst old_internal(GLIBC _2.2) [1]</code>	<code>mbstowes(GLIBC_ 2.2)mbstowcs(GLIB C_2.2) [2]</code>	<code>wertomb(GLIBC_2. 2)wrtomb(GLIBC_ 2.2) [2]</code>	<code>wesrehr(GLIBC_2.2)wscrchr(GLIBC_2. 2) [2]</code>	<code>weswidth(GLIBC_2 2)wcswidth(GLIBC _2.2) [2]</code>
<code>__westoul_internal(GLIBC_2.2) __wcst oul_internal(GLIBC _2.2) [1]</code>	<code>mbtowe(GLIBC_2. 2)mbtowc(GLIBC_ 2.2) [2]</code>	<code>wescasecmp(GLIB C_2.2)wscasecmp(GLIBC_2.2) [1]</code>	<code>wesrtombs(GLIBC_ 2.2)wscrombs(GLI BC_2.2) [2]</code>	<code>wesxfrm(GLIBC_2. 2)wcsxfrm(GLIBC_ 2.2) [2]</code>
<code>btowe(GLIBC_2.2) btowc(GLIBC_2.2) [2]</code>	<code>putwe(GLIBC_2.2) putwc(GLIBC_2.2) [2]</code>	<code>wesecat(GLIBC_2.2) wscat(GLIBC_2.2) [2]</code>	<code>wesspn(GLIBC_2.2)wcsspn(GLIBC_2. 2) [2]</code>	<code>wetob(GLIBC_2.2) wctob(GLIBC_2.2) [2]</code>
<code>fgetwe(GLIBC_2.2) fgetwc(GLIBC_2.2) [2]</code>	<code>putwehar(GLIBC_2 2)putwchar(GLIBC _2.2) [2]</code>	<code>weschr(GLIBC_2.2) wchr(GLIBC_2.2) [2]</code>	<code>wesstr(GLIBC_2.2) wcsstr(GLIBC_2.2) [2]</code>	<code>wetomb(GLIBC_2. 2)wctomb(GLIBC_ 2.2) [2]</code>
<code>fgetws(GLIBC_2.2) fgetws(GLIBC_2.2) [2]</code>	<code>swprintf(GLIBC_2. 2)swprintf(GLIBC_ 2.2) [2]</code>	<code>wesemp(GLIBC_2. 2)wscmp(GLIBC_ 2.2) [2]</code>	<code>westod(GLIBC_2.2) westod(GLIBC_2.2) [2]</code>	<code>wetrans(GLIBC_2.2)wctrans(GLIBC_2. 2) [2]</code>
<code>fputwe(GLIBC_2.2) fputwc(GLIBC_2.2) [2]</code>	<code>swscanf(GLIBC_2. 2)swscanf(GLIBC_ 2.2) [2]</code>	<code>wescoll(GLIBC_2.2)wscoll(GLIBC_2. 2) [2]</code>	<code>westof(GLIBC_2.2) westof(GLIBC_2.2) [2]</code>	<code>wetype(GLIBC_2.2)wctype(GLIBC_2. 2) [2]</code>
<code>fputws(GLIBC_2.2) fputws(GLIBC_2.2) [2]</code>	<code>towetrans(GLIBC_2 2)towctrans(GLIB C_2.2) [2]</code>	<code>wesepy(GLIBC_2.2)wscpy(GLIBC_2. 2) [2]</code>	<code>westoimax(GLIBC_ 2.2)wcstoimax(GLI BC_2.2) [2]</code>	<code>wewidth(GLIBC_2. 2)wcwidth(GLIBC_ 2.2) [2]</code>
<code>fwide(GLIBC_2.2)f wide(GLIBC_2.2) [2]</code>	<code>tolower(GLIBC_2 2)tolower(GLIBC _2.2) [2]</code>	<code>wesespn(GLIBC_2. 2)wscspn(GLIBC_ 2.2) [2]</code>	<code>westok(GLIBC_2.2) westok(GLIBC_2.2) [2]</code>	<code>wmemchr(GLIBC_ 2.2)wmemchr(GLIB C_2.2) [2]</code>
<code>fwprintf(GLIBC_2. 2)fwprintf(GLIBC_ 2.2) [2]</code>	<code>towupper(GLIBC_2 2)towupper(GLIBC _2.2) [2]</code>	<code>wesdup(GLIBC_2.2)wcdup(GLIBC_2. 2) [1]</code>	<code>westol(GLIBC_2.2) westol(GLIBC_2.2) [2]</code>	<code>wmemcmp(GLIBC_ _2.2)wmemcmp(GL IBC_2.2) [2]</code>
<code>fwscanf(GLIBC_2.2)fwscanf(GLIBC_2. 2) [2]</code>	<code>ungetwe(GLIBC_2. 2)ungetwc(GLIBC_ 2.2) [2]</code>	<code>wesftime(GLIBC_2. 2)wscftime(GLIBC _2.2) [2]</code>	<code>westold(GLIBC_2.2)wcstold(GLIBC_2. 2) [2]</code>	<code>wmemcpy(GLIBC_ 2.2)wmemcpy(GLI BC_2.2) [2]</code>
<code>getwe(GLIBC_2.2) getwc(GLIBC_2.2) [2]</code>	<code>vfwprintf(GLIBC_2 2)vfwprintf(GLIBC _2.2) [2]</code>	<code>weslen(GLIBC_2.2) wscnlen(GLIBC_2.2) [2]</code>	<code>westoll(GLIBC_2.2)wcstoll(GLIBC_2. 2) [2]</code>	<code>wmemmove(GLIB C_2.2)wmemmove(GLIBC_2.2) [2]</code>
<code>getwehar(GLIBC_2. 2)getwchar(GLIBC _2.2) [2]</code>	<code>vfwscanf(GLIBC_2. 2)vfwscanf(GLIBC _2.2) [2]</code>	<code>wesncasecmp(GLIB C_2.2)wscncasecmp (GLIBC_2.2) [1]</code>	<code>westombs(GLIBC_ 2.2)wcstombs(GLIB C_2.2) [2]</code>	<code>wmemset(GLIBC_2 2)wmemset(GLIBC _2.2) [2]</code>
<code>mblen(GLIBC_2.2) mblen(GLIBC_2.2)</code>	<code>vswprintf(GLIBC_2 2)vswprintf(GLIBC</code>	<code>wesncat(GLIBC_2. 2)wscncat(GLIBC_</code>	<code>westoq(GLIBC_2.2) westoq(GLIBC_2.2)</code>	<code>wprintf(GLIBC_2.2)wprintf(GLIBC_2.</code>

[2]	_2.2) [2]	2.2) [2]	[1]	2) [2]
mbrlen(GLIBC_2.2) mbrlen(GLIBC_2.2) [2]	vswscanf(GLIBC_2.2) vswscanf(GLIBC_2.2) [2]	wesnemp(GLIBC_2.2) wcsncmp(GLIBC_2.2) [2]	westoul(GLIBC_2.2) wcstoul(GLIBC_2.2) [2]	wscanf(GLIBC_2.2) wscanf(GLIBC_2.2) [2]
mbrtowc(GLIBC_2.2) mbrtowc(GLIBC_2.2) [2]	vwprintf(GLIBC_2.2) vwprintf(GLIBC_2.2) [2]	wesnecpy(GLIBC_2.2) wcsncpy(GLIBC_2.2) [2]	westoull(GLIBC_2.2) wcstoull(GLIBC_2.2) [2]	

103

104 *Referenced Specification(s)*

105 [1]. Linux Standard Base this specification

106 [2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)

107 V3)

1.2.8. String Functions

1.2.8.1. Interfaces for String Functions

109 An LSB conforming implementation shall provide the architecture specific functions for String Functions specified in
110 Table 1-13, with the full functionality as described in the referenced underlying specification.

111 **Table 1-13. libc - String Functions Function Interfaces**

__mempcpy(GLIBC_2.2) __mempcpy(GLIBC_2.2) [1]	bzero(GLIBC_2.2) bzero(GLIBC_2.2) [2]	streasestr(GLIBC_2.2) strcasestr(GLIBC_2.2) [1]	strncasecmp(GLIBC_2.2) strncasecmp(GLIBC_2.2) [2]	strtoimax(GLIBC_2.2) strtoimax(GLIBC_2.2) [2]
__rawmemchr(GLIBC_2.2) __rawmemchr(GLIBC_2.2) [1]	ffs(GLIBC_2.2) ffs(GLIBC_2.2) [2]	streat(GLIBC_2.2) strcat(GLIBC_2.2) [2]	strncat(GLIBC_2.2) strncat(GLIBC_2.2) [2]	strtok(GLIBC_2.2) strtok(GLIBC_2.2) [2]
__stpcpy(GLIBC_2.2) __stpcpy(GLIBC_2.2) [1]	index(GLIBC_2.2) index(GLIBC_2.2) [2]	strchr(GLIBC_2.2) strchr(GLIBC_2.2) [2]	strncmp(GLIBC_2.2) strncmp(GLIBC_2.2) [2]	strtok_r(GLIBC_2.2) strtok_r(GLIBC_2.2) [4+2]
__strdup(GLIBC_2.2) __strdup(GLIBC_2.2) [1]	memcpy(GLIBC_2.2) memcpy(GLIBC_2.2) [2]	strcmp(GLIBC_2.2) strcmp(GLIBC_2.2) [2]	strncpy(GLIBC_2.2) strncpy(GLIBC_2.2) [2]	strtol(GLIBC_2.2) strtol(GLIBC_2.2) [2]
__strtod_internal(GLIBC_2.2) __strtod_internal(GLIBC_2.2) [1]	memchr(GLIBC_2.2) memchr(GLIBC_2.2) [2]	streq(GLIBC_2.2) strcoll(GLIBC_2.2) [2]	strndup(GLIBC_2.2) strndup(GLIBC_2.2) [1]	strtoll(GLIBC_2.2) strtoll(GLIBC_2.2) [2]
__strtof_internal(GLIBC_2.2) __strtof_internal(GLIBC_2.2) [1]	memcmp(GLIBC_2.2) memcmp(GLIBC_2.2) [2]	streq(GLIBC_2.2) strcp(GLIBC_2.2) [2]	strnlen(GLIBC_2.2) strnlen(GLIBC_2.2) [1]	strtoq(GLIBC_2.2) strtoq(GLIBC_2.2) [1]
__strtok_r(GLIBC_2.2)	memcpy(GLIBC_2.2)	strespn(GLIBC_2.2)	strpbrk(GLIBC_2.2)	strtoull(GLIBC_2.2)

<code>__strtok_r(GLIBC_2.2)</code> [1]	<code>memcpy(GLIBC_2.2)</code> [2]	<code>strcspn(GLIBC_2.2)</code> [2]	<code>strpbrk(GLIBC_2.2)</code> [2]	<code>strtoull(GLIBC_2.2)</code> [2]
<code>__strtol_internal(GLIBC_2.2)</code> [1]	<code>memmove(GLIBC_2.2)</code> [2]	<code>strdup(GLIBC_2.2)</code> [2]	<code>strptime(GLIBC_2.2)</code> [1]	<code>strtoumax(GLIBC_2.2)</code> [2]
<code>__strtold_internal(GLIBC_2.2)</code> [1]	<code>memrchr(GLIBC_2.2)</code> [1]	<code>strerror(GLIBC_2.2)</code> [2]	<code>strchr(GLIBC_2.2)</code> [2]	<code>strtouq(GLIBC_2.2)</code> [1]
<code>__strtoll_internal(GLIBC_2.2)</code> [1]	<code>memset(GLIBC_2.2)</code> [2]	<code>strerror_r(GLIBC_2.2)</code> [1]	<code>strsep(GLIBC_2.2)</code> [1]	<code>strverscmp(GLIBC_2.2)</code> [1]
<code>__strtoul_internal(GLIBC_2.2)</code> [1]	<code>rindex(GLIBC_2.2)</code> [2]	<code>strfmon(GLIBC_2.2)</code> [2]	<code>strsignal(GLIBC_2.2)</code> [1]	<code>strxfrm(GLIBC_2.2)</code> [2]
<code>__strtoull_internal(GLIBC_2.2)</code> [1]	<code>stpcpy(GLIBC_2.2)</code> [1]	<code>strfry(GLIBC_2.2)</code> [1]	<code>strspn(GLIBC_2.2)</code> [2]	<code>swab(GLIBC_2.2)</code> [2]
<code>bcmp(GLIBC_2.2)</code> [2]	<code>stpncpy(GLIBC_2.2)</code> [1]	<code>strftime(GLIBC_2.2)</code> [2]	<code>strstr(GLIBC_2.2)</code> [2]	
<code>bcopy(GLIBC_2.2)</code> [2]	<code>strcasecmp(GLIBC_2.2)</code> [2]	<code>strlen(GLIBC_2.2)</code> [2]	<code>strtof(GLIBC_2.2)</code> [2]	

112

113 *Referenced Specification(s)*114 [1]. *Linux Standard Base* this specification115 [2]. *ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)*116 *Vol 3*)

1.2.9. IPC Functions

1.2.9.1. Interfaces for IPC Functions

118 An LSB conforming implementation shall provide the architecture specific functions for IPC Functions specified in
119 Table 1-14, with the full functionality as described in the referenced underlying specification.

120 **Table 1-14. libc - IPC Functions Function Interfaces**

<code>ftok(GLIBC_2.2)</code>	<code>msgrev(GLIBC_2.2)</code>	<code>semget(GLIBC_2.2)</code>	<code>shmetl(GLIBC_2.2)</code>	
------------------------------	--------------------------------	--------------------------------	--------------------------------	--

	k(GLIBC_2.2) [1]	̀msgrcv(GLIBC_2.2) [1]	semget(GLIBC_2.2) [1]	shmctl(GLIBC_2.2) [1]	
	msgctl(GLIBC_2.2) [1]	msgsnd(GLIBC_2.2) [1]	semop(GLIBC_2.2) [1]	shmdt(GLIBC_2.2) [1]	
121	msgget(GLIBC_2.2) [1]	semctl(GLIBC_2.2) [1]	shmat(GLIBC_2.2) [1]	shmget(GLIBC_2.2) [1]	

122 *Referenced Specification(s)*

123 [1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
124 V3)

1.2.10. Regular Expressions

1.2.10.1. Interfaces for Regular Expressions

126 An LSB conforming implementation shall provide the architecture specific functions for Regular Expressions
127 specified in Table 1-15, with the full functionality as described in the referenced underlying specification.

128 **Table 1-15. libc - Regular Expressions Function Interfaces**

	regcomp(GLIBC_2.2) [1]	regerror(GLIBC_2.2) [1]	regexec(GLIBC_2.2) [1]	regfree(GLIBC_2.2) [1]	
--	------------------------	-------------------------	------------------------	------------------------	--

130 *Referenced Specification(s)*

131 [1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
132 V3)

133 An LSB conforming implementation shall provide the architecture specific deprecated functions for Regular
134 Expressions specified in Table 1-16, with the full functionality as described in the referenced underlying specification.

135 These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn
136 in future releases of this specification.

137 **Table 1-16. libc - Regular Expressions Deprecated Function Interfaces**

	advance(GLIBC_2.2) [1]	re_comp(GLIBC_2.2) [1]	re_exec(GLIBC_2.2) [1]	step(GLIBC_2.2) [1]	
--	------------------------	------------------------	------------------------	---------------------	--

139 *Referenced Specification(s)*

140 [1]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,
141 C606) SUSv2

142 An LSB conforming implementation shall provide the architecture specific deprecated data interfaces for Regular
143 Expressions specified in Table 1-17, with the full functionality as described in the referenced underlying specification.

144 These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn
 145 in future releases of this specification.

146 **Table 1-17. libc - Regular Expressions Deprecated Data Interfaces**

147	<code>loe1(GLIBC_2.2)lo c1(GLIBC_2.2) [1]</code>	<code>loe2(GLIBC_2.2)lo c2(GLIBC_2.2) [1]</code>	<code>loes(GLIBC_2.2)loc s(GLIBC_2.2) [1]</code>		
-----	------------------------------------------------------	------------------------------------------------------	------------------------------------------------------	--	--

148 *Referenced Specification(s)*

149 [1]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,
 150 €606)SUSv2

1.2.11. Character Type Functions

1.2.11.1. Interfaces for Character Type Functions

152 An LSB conforming implementation shall provide the architecture specific functions for Character Type Functions
 153 specified in Table 1-18, with the full functionality as described in the referenced underlying specification.

154 **Table 1-18. libc - Character Type Functions Function Interfaces**

155	<code>__ctype_get_mb_cu r_max(GLIBC_2.2) __ctype_get_mb_cu r_max(GLIBC_2.2) [1]</code>	<code>isdigit(GLIBC_2.2)i sdigit(GLIBC_2.2) [2]</code>	<code>iswalnum(GLIBC_2. 2)iswalnum(GLIB C_2.2) [2]</code>	<code>iswlower(GLIBC_2. 2)iswlower(GLIBC _2.2) [2]</code>	<code>toascii(GLIBC_2.2) toascii(GLIBC_2.2) [2]</code>
	<code>_tolower(GLIBC_2. 2)_tolower(GLIBC_ 2.2) [2]</code>	<code>isgraph(GLIBC_2.2) isgraph(GLIBC_2. 2) [2]</code>	<code>iswalpha(GLIBC_2. 2)iswalpha(GLIBC_ 2.2) [2]</code>	<code>iswprint(GLIBC_2. 2)iswprint(GLIBC_ 2.2) [2]</code>	<code>tolower(GLIBC_2.2) tolower(GLIBC_2. 2) [2]</code>
	<code>_toupper(GLIBC_2. 2)_toupper(GLIBC_ 2.2) [2]</code>	<code>islower(GLIBC_2.2) islower(GLIBC_2. 2) [2]</code>	<code>iswblank(GLIBC_2. 2)iswblank(GLIBC _2.2) [2]</code>	<code>iswpunct(GLIBC_2. 2)iswpunct(GLIBC _2.2) [2]</code>	<code>toupper(GLIBC_2.2) toupper(GLIBC_2. 2) [2]</code>
	<code>isalnum(GLIBC_2.2) isalnum(GLIBC_2. 2) [2]</code>	<code>isprint(GLIBC_2.2)i sprint(GLIBC_2.2) [2]</code>	<code>iswcntrl(GLIBC_2. 2)iswcntrl(GLIBC_ 2.2) [2]</code>	<code>iswspace(GLIBC_2. 2)iswspace(GLIBC _2.2) [2]</code>	
	<code>isalpha(GLIBC_2.2) isalpha(GLIBC_2.2) [2]</code>	<code>ispunct(GLIBC_2.2) ispunct(GLIBC_2. 2) [2]</code>	<code>iswctype(GLIBC_2. 2)iswctype(GLIBC_ 2.2) [1+2]</code>	<code>iswupper(GLIBC_2. 2)iswupper(GLIBC _2.2) [2]</code>	
	<code>isascii(GLIBC_2.2)i sascii(GLIBC_2.2) [2]</code>	<code>isspace(GLIBC_2.2) isspace(GLIBC_2. 2) [2]</code>	<code>iswdigit(GLIBC_2. 2)iswdigit(GLIBC_ 2.2) [2]</code>	<code>iswxdigit(GLIBC_2. 2)iswxdigit(GLIBC _2.2) [2]</code>	
	<code>isctrl(GLIBC_2.2)i sctrl(GLIBC_2.2) [2]</code>	<code>isupper(GLIBC_2.2) isupper(GLIBC_2. 2) [2]</code>	<code>iswgraph(GLIBC_2. 2)iswgraph(GLIBC _2.2) [2]</code>	<code>isxdigit(GLIBC_2.2) isxdigit(GLIBC_2. 2) [2]</code>	

156 *Referenced Specification(s)*

157 [1]. ~~Linux Standard Base~~this specification

158 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)~~

159 ~~∇3)~~

1.2.12. Time Manipulation

1.2.12.1. Interfaces for Time Manipulation

161 An LSB conforming implementation shall provide the architecture specific functions for Time Manipulation specified
162 in Table 1-19, with the full functionality as described in the referenced underlying specification.

163 **Table 1-19. libc - Time Manipulation Function Interfaces**

adjtime(GLIBC_2.2)adjtime(GLIBC_2. 2) [1]	etime(GLIBC_2.2)c time(GLIBC_2.2) [2]	gmtime(GLIBC_2.2)gmtime(GLIBC_2. 2) [2]	localtime_r(GLIBC _2.2)localtime_r(G LIBC_2.2) [2]	ualarm(GLIBC_2.2) ualarm(GLIBC_2.2) [2]
asctime(GLIBC_2.2)asctime(GLIBC_2. 2) [2]	etime_r(GLIBC_2.2)ctime_r(GLIBC_2. 2) [2]	gmtime_r(GLIBC_2 .2)gmtime_r(GLIB C_2.2) [2]	mktime(GLIBC_2.2)mktime(GLIBC_2. 2) [2]	
asctime_r(GLIBC_2 .2)asctime_r(GLIB C_2.2) [2]	difftime(GLIBC_2. 2)difftime(GLIBC_ 2.2) [2]	localtime(GLIBC_2)localtime(GLIBC _2.2) [2]	tzset(GLIBC_2.2)tz set(GLIBC_2.2) [2]	

165 *Referenced Specification(s)*

166 [1]. ~~Linux Standard Base~~this specification

167 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)~~

168 ~~∇3)~~

169 An LSB conforming implementation shall provide the architecture specific deprecated functions for Time
170 Manipulation specified in Table 1-20, with the full functionality as described in the referenced underlying
171 specification.

172 These interfaces are deprecated, and applications should avoid using them. These interfaces may be withdrawn
173 in future releases of this specification.

174 **Table 1-20. libc - Time Manipulation Deprecated Function Interfaces**

adjtimex(GLIBC_2. 2)adjtimex(GLIBC_ 2.2) [1]				
-----------------------------------------------------------------	--	--	--	--

176 *Referenced Specification(s)*

177 [1]. ~~Linux Standard Base~~this specification

178 An LSB conforming implementation shall provide the architecture specific data interfaces for Time Manipulation
179 specified in Table 1-21, with the full functionality as described in the referenced underlying specification.

180 **Table 1-21. libc - Time Manipulation Data Interfaces**

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

182 *Referenced Specification(s)*183 [1]. ~~Linux Standard Base~~this specification184 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

1.2.13. Terminal Interface Functions

1.2.13.1. Interfaces for Terminal Interface Functions

187 An LSB conforming implementation shall provide the architecture specific functions for Terminal Interface Functions
188 specified in Table 1-22, with the full functionality as described in the referenced underlying specification.

189 **Table 1-22. libc - Terminal Interface Functions Function Interfaces**

efgetispeed(GLIBC_2.2) cfgetispeed(GLIBC_2.2) [1]	efsetispeed(GLIBC_2.2) cfsetispeed(GLIBC_2.2) [1]	tedrain(GLIBC_2.2) tcdrain(GLIBC_2.2) [1]	tegetattr(GLIBC_2.2) tcgetattr(GLIBC_2.2) [1]	tesendbreak(GLIBC_2.2) tcsendbreak(GLIBC_2.2) [1]
efgetospeed(GLIBC_2.2) cfgetospeed(GLIBC_2.2) [1]	efsetospeed(GLIBC_2.2) cfsetospeed(GLIBC_2.2) [1]	teflow(GLIBC_2.2) cflow(GLIBC_2.2) [1]	tegetpgrp(GLIBC_2.2) tcgetpgrp(GLIBC_2.2) [1]	tesetattr(GLIBC_2.2) tcsetattr(GLIBC_2.2) [1]
efmakeraw(GLIBC_2.2) cfmakeraw(GLIBC_2.2) [2]	efsetspeed(GLIBC_2.2) cfsetspeed(GLIBC_2.2) [2]	teflush(GLIBC_2.2) tcflush(GLIBC_2.2) [1]	tegetsid(GLIBC_2.2) tcgetsid(GLIBC_2.2) [1]	tesetpgrp(GLIBC_2.2) tcsetpgrp(GLIBC_2.2) [1]

191 *Referenced Specification(s)*192 [1]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~194 [2]. ~~Linux Standard Base~~this specification

1.2.14. System Database Interface

1.2.14.1. Interfaces for System Database Interface

196 An LSB conforming implementation shall provide the architecture specific functions for System Database Interface
197 specified in Table 1-23, with the full functionality as described in the referenced underlying specification.

198 **Table 1-23. libc - System Database Interface Function Interfaces**

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

199

200 *Referenced Specification(s)*201 [1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
202 V3)

203 [2]. Linux Standard Base this specification

204 [3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,
205 C606) SUSv2**1.2.15. Language Support**206 **1.2.15.1. Interfaces for Language Support**207 An LSB conforming implementation shall provide the architecture specific functions for Language Support specified
208 in Table 1-24, with the full functionality as described in the referenced underlying specification.

209 **Table 1-24. libc - Language Support Function Interfaces**

<code>__libc_start_main(GLIBC_2.2) __libc_ start_main(GLIBC_ 2.2) [1]</code>	<code>__obstack_begin(GLI IBC_2.2) __obstack_ begin(GLIBC_2.2) [1]</code>	<code>__obstack_newchunk (GLIBC_2.2) __obsta ck_newchunk(GLIB C_2.2) [1]</code>	<code>obstack_free(GLIB C_2.2) obstack_free(GLIBC_2.2) [1]</code>	
-------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------	----------------------------------------------------------------------------	--

211 *Referenced Specification(s)*212 [1]. ~~Linux Standard Base~~this specification

1.2.16. Large File Support

1.2.16.1. Interfaces for Large File Support

214 An LSB conforming implementation shall provide the architecture specific functions for Large File Support specified
215 in Table 1-25, with the full functionality as described in the referenced underlying specification.

216 **Table 1-25. libc - Large File Support Function Interfaces**

<code>__fxstat64(GLIBC_ 2.2) __fxstat64(GLI BC_2.2) [1]</code>	<code>fopen64(GLIBC_2. 2)fopen64(GLIBC_ 2.2) [2]</code>	<code>ftello64(GLIBC_2.2)ftello64(GLIBC_2. 2) [2]</code>	<code>lseek64(GLIBC_2.2)lseek64(GLIBC_2. 2) [2]</code>	<code>readdir64(GLIBC_2)readdir64(GLIBC _2.2) [2]</code>
<code>__lxstat64(GLIBC_ 2.2) __lxstat64(GLI BC_2.2) [1]</code>	<code>freopen64(GLIBC_ 2.2)freopen64(GLI BC_2.2) [2]</code>	<code>ftruncate64(GLIBC _2.2)ftruncate64(G LIBC_2.2) [2]</code>	<code>mkstemp64(GLIBC _2.2)mkstemp64(G LIBC_2.2) [2]</code>	<code>statvfs64(GLIBC_2. 2)statvfs64(GLIBC _2.2) [2]</code>
<code>__xstat64(GLIBC_ 2.2) __xstat64(GLIB C_2.2) [1]</code>	<code>fseeko64(GLIBC_2. 2)fseeko64(GLIBC _2.2) [2]</code>	<code>ftw64(GLIBC_2.2)f tw64(GLIBC_2.2) [2]</code>	<code>mmap64(GLIBC_2. 2)mmap64(GLIBC_ 2.2) [2]</code>	<code>tmpfile64(GLIBC_2)tmpfile64(GLIB C_2.2) [2]</code>
<code>creat64(GLIBC_2.2)creat64(GLIBC_2. 2) [2]</code>	<code>fsetpos64(GLIBC_2)fsetpos64(GLIBC _2.2) [2]</code>	<code>getrlimit64(GLIBC _2.2)getrlimit64(GL IBC_2.2) [2]</code>	<code>nftw64(GLIBC_2.2) nftw64(GLIBC_2.2) [2]</code>	<code>truncate64(GLIBC_ 2.2)truncate64(GLI BC_2.2) [2]</code>
<code>fgetpos64(GLIBC_ 2.2)fgetpos64(GLIB C_2.2) [2]</code>	<code>fstatvfs64(GLIBC_ 2.2)fstatvfs64(GLIB C_2.2) [2]</code>	<code>lockf64(GLIBC_2.2)lockf64(GLIBC_2. 2) [2]</code>	<code>open64(GLIBC_2.2)open64(GLIBC_2. 2) [2]</code>	

218 *Referenced Specification(s)*219 [1]. ~~Linux Standard Base~~this specification

220 [2]. Large File Support

1.2.17. Standard Library

1.2.17.1. Interfaces for Standard Library

222 An LSB conforming implementation shall provide the architecture specific functions for Standard Library specified in
223 Table 1-26, with the full functionality as described in the referenced underlying specification.

Table 1-26. libc - Standard Library Function Interfaces

<code>_Exit(GLIBC_2.2)</code> <code>Exit(GLIBC_2.2)</code> [1]	<code>dirname(GLIBC_2.2)</code> <code>dirname(GLIBC_2.2)</code> [1]	<code>glob(GLIBC_2.2)</code> <code>glob(GLIBC_2.2)</code> [1]	<code>lsearch(GLIBC_2.2)</code> <code>lsearch(GLIBC_2.2)</code> [1]	<code>srand(GLIBC_2.2)</code> <code>srand(GLIBC_2.2)</code> [1]
<code>__assert_fail(GLIBC_2.2)</code> <code>__assert_fail(GLIBC_2.2)</code> [2]	<code>div(GLIBC_2.2)</code> <code>div(GLIBC_2.2)</code> [1]	<code>glob64(GLIBC_2.2)</code> <code>glob64(GLIBC_2.2)</code> [2]	<code>makecontext(GLIBC_2.2)</code> <code>makecontext(GLIBC_2.2)</code> [1]	<code>srand48(GLIBC_2.2)</code> <code>rand48(GLIBC_2.2)</code> [1]
<code>__cxa_atexit(GLIBC_2.2)</code> <code>__cxa_atexit(GLIBC_2.2)</code> [2]	<code>drand48(GLIBC_2.2)</code> <code>drand48(GLIBC_2.2)</code> [1]	<code>globfree(GLIBC_2.2)</code> <code>globfree(GLIBC_2.2)</code> [1]	<code>malloc(GLIBC_2.2)</code> <code>malloc(GLIBC_2.2)</code> [1]	<code>srandom(GLIBC_2.2)</code> <code>srandom(GLIBC_2.2)</code> [1]
<code>__errno_location(GLIBC_2.2)</code> <code>__errno_location(GLIBC_2.2)</code> [2]	<code>ecvt(GLIBC_2.2)</code> <code>ecvt(GLIBC_2.2)</code> [1]	<code>globfree64(GLIBC_2.2)</code> <code>globfree64(GLIBC_2.2)</code> [2]	<code>memmem(GLIBC_2.2)</code> <code>memmem(GLIBC_2.2)</code> [2]	<code>strtod(GLIBC_2.2)</code> <code>strtod(GLIBC_2.2)</code> [1]
<code>__fpending(GLIBC_2.2)</code> <code>__fpending(GLIBC_2.2)</code> [2]	<code>erand48(GLIBC_2.2)</code> <code>erand48(GLIBC_2.2)</code> [1]	<code>grantpt(GLIBC_2.2)</code> <code>grantpt(GLIBC_2.2)</code> [1]	<code>mkstemp(GLIBC_2.2)</code> <code>mkstemp(GLIBC_2.2)</code> [1]	<code>strtol(GLIBC_2.2)</code> <code>strtol(GLIBC_2.2)</code> [1]
<code>__getpagesize(GLIBC_2.2)</code> <code>__getpagesize(GLIBC_2.2)</code> [2]	<code>err(GLIBC_2.2)</code> <code>err(GLIBC_2.2)</code> [2]	<code>hcreate(GLIBC_2.2)</code> <code>hcreate(GLIBC_2.2)</code> [1]	<code>mktemp(GLIBC_2.2)</code> <code>mktemp(GLIBC_2.2)</code> [1]	<code>strtoul(GLIBC_2.2)</code> <code>strtoul(GLIBC_2.2)</code> [1]
<code>__isinf(GLIBC_2.2)</code> <code>__isinf(GLIBC_2.2)</code> [2]	<code>error(GLIBC_2.2)</code> <code>error(GLIBC_2.2)</code> [2]	<code>hdestroy(GLIBC_2.2)</code> <code>hdestroy(GLIBC_2.2)</code> [1]	<code>mrand48(GLIBC_2.2)</code> <code>mrand48(GLIBC_2.2)</code> [1]	<code>swapecontext(GLIBC_2.2)</code> <code>swapcontext(GLIBC_2.2)</code> [1]
<code>__isinff(GLIBC_2.2)</code> <code>__isinff(GLIBC_2.2)</code> [2]	<code>errx(GLIBC_2.2)</code> <code>errx(GLIBC_2.2)</code> [2]	<code>hsearch(GLIBC_2.2)</code> <code>hsearch(GLIBC_2.2)</code> [1]	<code>nftw(GLIBC_2.2)</code> <code>nftw(GLIBC_2.2)</code> [1]	<code>syslog(GLIBC_2.2)</code> <code>syslog(GLIBC_2.2)</code> [1]
<code>__isinfl(GLIBC_2.2)</code> <code>__isinfl(GLIBC_2.2)</code> [2]	<code>fevt(GLIBC_2.2)</code> <code>fevt(GLIBC_2.2)</code> [1]	<code>htonl(GLIBC_2.2)</code> <code>htonl(GLIBC_2.2)</code> [1]	<code>nrand48(GLIBC_2.2)</code> <code>nrand48(GLIBC_2.2)</code> [1]	<code>system(GLIBC_2.2)</code> <code>system(GLIBC_2.2)</code> [2]
<code>__isnan(GLIBC_2.2)</code> <code>__isnan(GLIBC_2.2)</code> [2]	<code>fmtmsg(GLIBC_2.2)</code> <code>fmtmsg(GLIBC_2.2)</code> [1]	<code>htons(GLIBC_2.2)</code> <code>htons(GLIBC_2.2)</code> [1]	<code>ntohl(GLIBC_2.2)</code> <code>ntohl(GLIBC_2.2)</code> [1]	<code>tdelete(GLIBC_2.2)</code> <code>tdelete(GLIBC_2.2)</code> [1]
<code>__isnanf(GLIBC_2.2)</code> <code>__isnanf(GLIBC_2.2)</code> [2]	<code>fnmatch(GLIBC_2.2)</code> <code>fnmatch(GLIBC_2.2)</code> [1]	<code>imaxabs(GLIBC_2.2)</code> <code>imaxabs(GLIBC_2.2)</code> [1]	<code>ntohs(GLIBC_2.2)</code> <code>ntohs(GLIBC_2.2)</code> [1]	<code>tfind(GLIBC_2.2)</code> <code>tfind(GLIBC_2.2)</code> [1]
<code>__isnank(GLIBC_2.2)</code> <code>__isnank(GLIBC_2.2)</code> [2]	<code>fpathconf(GLIBC_2.2)</code> <code>fpathconf(GLIBC_2.2)</code> [1]	<code>imaxdiv(GLIBC_2.2)</code> <code>imaxdiv(GLIBC_2.2)</code> [1]	<code>openlog(GLIBC_2.2)</code> <code>openlog(GLIBC_2.2)</code> [1]	<code>tmpfile(GLIBC_2.2)</code> <code>tmpfile(GLIBC_2.2)</code> [1]
<code>__sysconf(GLIBC_2.2)</code> <code>__sysconf(GLIBC_2.2)</code> [1]	<code>free(GLIBC_2.2)</code> <code>free(GLIBC_2.2)</code> [1]	<code>inet_addr(GLIBC_2.2)</code> <code>inet_addr(GLIBC_2.2)</code> [1]	<code>perror(GLIBC_2.2)</code> <code>perror(GLIBC_2.2)</code> [1]	<code>tmpnam(GLIBC_2.2)</code> <code>tmpnam(GLIBC_2.2)</code> [1]

BC_2.2) [2]		_2.2) [1]	[1]	2.2) [1]
_exit(GLIBC_2.2) exit(GLIBC_2.2) [1]	freeaddrinfo(GLIBC_2.2) freeaddrinfo(GLIBC_2.2) [1]	inet_ntoa(GLIBC_2.2) inet_ntoa(GLIBC_2.2) [1]	posix_memalign(GLIBC_2.2) posix_memalign(GLIBC_2.2) [1]	tsearch(GLIBC_2.2) tsearch(GLIBC_2.2) [1]
_longjmp(GLIBC_2.2) _longjmp(GLIBC_2.2) [1]	ftw(GLIBC_2.2) ftw(GLIBC_2.2) [1]	inet_ntop(GLIBC_2.2) inet_ntop(GLIBC_2.2) [1]	ptsname(GLIBC_2.2) ptsname(GLIBC_2.2) [1]	ttynamename(GLIBC_2.2) ttynamename(GLIBC_2.2) [1]
_setjmp(GLIBC_2.2) _setjmp(GLIBC_2.2) [1]	ftw(GLIBC_2.2) ftw(GLIBC_2.2) [1]	inet_pton(GLIBC_2.2) inet_pton(GLIBC_2.2) [1]	putenv(GLIBC_2.2) putenv(GLIBC_2.2) [1]	ttynamename_r(GLIBC_2.2) ttynamename_r(GLIBC_2.2) [1]
a64(GLIBC_2.2) a64(GLIBC_2.2) [1]	funlockfile(GLIBC_2.2) funlockfile(GLIBC_2.2) [1]	initstate(GLIBC_2.2) initstate(GLIBC_2.2) [1]	qsort(GLIBC_2.2) qsort(GLIBC_2.2) [1]	twalk(GLIBC_2.2) twalk(GLIBC_2.2) [1]
abort(GLIBC_2.2) abort(GLIBC_2.2) [1]	gai_strerror(GLIBC_2.2) gai_strerror(GLIBC_2.2) [1]	insque(GLIBC_2.2) insque(GLIBC_2.2) [1]	rand(GLIBC_2.2) rand(GLIBC_2.2) [1]	unlockpt(GLIBC_2.2) unlockpt(GLIBC_2.2) [1]
abs(GLIBC_2.2) abs(GLIBC_2.2) [1]	gevt(GLIBC_2.2) gevt(GLIBC_2.2) [1]	isatty(GLIBC_2.2) isatty(GLIBC_2.2) [1]	rand_r(GLIBC_2.2) rand_r(GLIBC_2.2) [1]	unsetenv(GLIBC_2.2) unsetenv(GLIBC_2.2) [1]
atof(GLIBC_2.2) atof(GLIBC_2.2) [1]	getaddrinfo(GLIBC_2.2) getaddrinfo(GLIBC_2.2) [1]	isblank(GLIBC_2.2) isblank(GLIBC_2.2) [1]	random(GLIBC_2.2) random(GLIBC_2.2) [1]	usleep(GLIBC_2.2) usleep(GLIBC_2.2) [1]
atoi(GLIBC_2.2) atoi(GLIBC_2.2) [1]	getcwd(GLIBC_2.2) getcwd(GLIBC_2.2) [1]	jrand48(GLIBC_2.2) jrand48(GLIBC_2.2) [1]	random_r(GLIBC_2.2) random_r(GLIBC_2.2) [2]	verrx(GLIBC_2.2) verrx(GLIBC_2.2) [2]
atol(GLIBC_2.2) atol(GLIBC_2.2) [1]	getdate(GLIBC_2.2) getdate(GLIBC_2.2) [1]	l64a(GLIBC_2.2) l64a(GLIBC_2.2) [1]	realloc(GLIBC_2.2) realloc(GLIBC_2.2) [1]	vfscanf(GLIBC_2.2) vfscanf(GLIBC_2.2) [1]
atoll(GLIBC_2.2) atoll(GLIBC_2.2) [1]	getenv(GLIBC_2.2) getenv(GLIBC_2.2) [1]	labs(GLIBC_2.2) labs(GLIBC_2.2) [1]	realpath(GLIBC_2.3) realpath(GLIBC_2.3) [1]	vscanf(GLIBC_2.2) vscanf(GLIBC_2.2) [1]
basename(GLIBC_2.2) basename(GLIBC_2.2) [1]	getlogin(GLIBC_2.2) getlogin(GLIBC_2.2) [1]	leong48(GLIBC_2.2) leong48(GLIBC_2.2) [1]	remque(GLIBC_2.2) remque(GLIBC_2.2) [1]	vsscanf(GLIBC_2.2) vsscanf(GLIBC_2.2) [1]
bsearch(GLIBC_2.2) bsearch(GLIBC_2.2) [1]	getnameinfo(GLIBC_2.2) getnameinfo(GLIBC_2.2) [1]	ldiv(GLIBC_2.2) ldiv(GLIBC_2.2) [1]	seed48(GLIBC_2.2) seed48(GLIBC_2.2) [1]	vsyslog(GLIBC_2.2) vsyslog(GLIBC_2.2) [2]
calloc(GLIBC_2.2) calloc(GLIBC_2.2)	getopt(GLIBC_2.2) getopt(GLIBC_2.2)	lfind(GLIBC_2.2) lfind(GLIBC_2.2) [1]	setenv(GLIBC_2.2) setenv(GLIBC_2.2)	warn(GLIBC_2.2) warn(GLIBC_2.2) [2]

[1]	[2]		[1]	
closelog(GLIBC_2.2)closelog(GLIBC_2.2) [1]	getopt_long(GLIBC_2.2)getopt_long(GLIBC_2.2) [2]	llabs(GLIBC_2.2)llabs(GLIBC_2.2) [1]	sethostid(GLIBC_2.2)sethostid(GLIBC_2.2) [2]	warnx(GLIBC_2.2)warnx(GLIBC_2.2) [2]
confstr(GLIBC_2.2)confstr(GLIBC_2.2) [1]	getopt_long_only(GLIBC_2.2)getopt_long_only(GLIBC_2.2) [2]	lldiv(GLIBC_2.2)lldiv(GLIBC_2.2) [1]	sethostname(GLIBC_2.2)sethostname(GLIBC_2.2) [2]	wordexp(GLIBC_2.2)wordexp(GLIBC_2.2) [1]
euserid(GLIBC_2.2)cuserid(GLIBC_2.2) [3]	getsubopt(GLIBC_2.2)getsubopt(GLIBC_2.2) [1]	longjmp(GLIBC_2.2)longjmp(GLIBC_2.2) [1]	setlogmask(GLIBC_2.2)setlogmask(GLIBC_2.2) [1]	wordfree(GLIBC_2.2)wordfree(GLIBC_2.2) [1]
daemon(GLIBC_2.2)daemon(GLIBC_2.2) [2]	gettimeofday(GLIBC_2.2)gettimeofday(GLIBC_2.2) [1]	lrand48(GLIBC_2.2)lrand48(GLIBC_2.2) [1]	setstate(GLIBC_2.2)setstate(GLIBC_2.2) [1]	

225

226 *Referenced Specification(s)*227 [1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS))
228 V3)

229 [2]. Linux Standard Base this specification

230 [3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,
231 C606) SUSv2232 An LSB conforming implementation shall provide the architecture specific data interfaces for Standard Library
233 specified in Table 1-27, with the full functionality as described in the referenced underlying specification.234 **Table 1-27. libc - Standard Library Data Interfaces**

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

235

236 *Referenced Specification(s)*

237 [1]. Linux Standard Base this specification

238 [2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS))
239 V3)

1.3. Data Definitions for libc

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

243 These definitions are intended to supplement those provided in the referenced underlying specifications.

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

1.3.1. errno.h

```
247
248 #define EDEADLOCK          EDEADLK
```

1.3.2. inttypes.h

```
249
250 typedef long intmax_t;
251 typedef unsigned long uintmax_t;
252 typedef unsigned long uintptr_t;
253 typedef unsigned long uint64_t;
```

1.3.3. limits.h

```
254
255 #define LONG_MAX           0x7FFFFFFFFFFFFFFFL
256 #define ULONG_MAX        0xFFFFFFFFFFFFFFFFUL
257
258 #define CHAR_MAX          SCHAR_MAX
259 #define CHAR_MIN          SCHAR_MIN
```

1.3.4. setjmp.h

```
260
261 typedef long __jmp_buf[70] __attribute__((aligned(16)));
```

1.3.5. signal.h

```
262
263 struct sigaction
264 {
265     union
266     {
267         sighandler_t _sa_handler;
268         void (*_sa_sigaction) (int, siginfo_t *, void *);
269     }
270     __sigaction_handler;
271     unsigned long sa_flags;
```

```

272     sigset_t sa_mask;
273 }
274 ;
275 #define MINSIGSTKSZ      131027
276 #define SIGSTKSZ        262144
277
278 struct ia64_fpreg
279 {
280     union
281     {
282         unsigned long bits[2];
283         long double __dummy;
284     }
285     u;
286 }
287 ;
288
289 struct sigcontext
290 {
291     unsigned long sc_flags;
292     unsigned long sc_nat;
293     stack_t sc_stack;
294     unsigned long sc_ip;
295     unsigned long sc_cfm;
296     unsigned long sc_um;
297     unsigned long sc_ar_rsc;
298     unsigned long sc_ar_bsp;
299     unsigned long sc_ar_rnat;
300     unsigned long sc_ar_ccv;
301     unsigned long sc_ar_unat;
302     unsigned long sc_ar_fpsr;
303     unsigned long sc_ar_pfs;
304     unsigned long sc_ar_lc;
305     unsigned long sc_pr;
306     unsigned long sc_br[8];
307     unsigned long sc_gr[32];
308     struct ia64_fpreg sc_fr[128];
309     unsigned long sc_rbs_base;
310     unsigned long sc_loadrs;
311     unsigned long sc_ar25;
312     unsigned long sc_ar26;
313     unsigned long sc_rsvd[12];
314     unsigned long sc_mask;
315 }
316 ;

```

1.3.6. stddef.h

```

317
318 typedef long ptrdiff_t;
319 typedef unsigned long size_t;

```

1.3.7. sys/ioctl.h

```

320
321 #define FIONREAD          0x541B
322 #define TIOCNOTTY       0x5422

```

1.3.8. sys/ipc.h

```

323
324 struct ipc_perm
325 {
326     key_t __key;
327     uid_t uid;
328     gid_t gid;
329     uid_t cuid;
330     uid_t cgid;
331     mode_t mode;
332     unsigned short __seq;
333     unsigned short __pad1;
334     unsigned long __unused1;
335     unsigned long __unused2;
336 }
337 ;

```

1.3.9. sys/mman.h

```

338
339 #define MCL_CURRENT      1
340 #define MCL_FUTURE      2

```

1.3.10. sys/msg.h

```

341
342 struct msqid_ds
343 {
344     struct ipc_perm msg_perm;
345     time_t msg_stime;
346     time_t msg_rtime;
347     time_t msg_ctime;
348     unsigned long __msg_cbytes;
349     unsigned long msg_qnum;
350     unsigned long msg_qbytes;
351     pid_t msg_lspid;
352     pid_t msg_lrpid;
353     unsigned long __unused1;
354     unsigned long __unused2;
355 }
356 ;

```

1.3.11. sys/sem.h

```

357
358 struct semid_ds
359 {
360     struct ipc_perm sem_perm;
361     time_t sem_otime;
362     time_t sem_ctime;
363     unsigned long sem_nsems;
364     unsigned long __unused1;
365     unsigned long __unused2;
366 }
367 ;

```

1.3.12. sys/shm.h

```

368
369 #define SHMLBA (1024*1024)
370
371 struct shmid_ds
372 {
373     struct ipc_perm shm_perm;
374     size_t shm_segsz;
375     time_t shm_atime;
376     time_t shm_dtime;
377     time_t shm_ctime;
378     pid_t shm_cpid;
379     pid_t shm_lpid;
380     unsigned long shm_nattch;
381     unsigned long __unused1;
382     unsigned long __unused2;
383 }
384 ;

```

1.3.13. sys/socket.h

```

385
386 typedef uint64_t __ss_aligntype;

```

1.3.14. sys/stat.h

```

387
388 #define _STAT_VER 1
389
390 struct stat
391 {
392     dev_t st_dev;
393     ino_t st_ino;
394     nlink_t st_nlink;
395     mode_t st_mode;
396     uid_t st_uid;

```



```

397     gid_t st_gid;
398     unsigned int pad0;
399     dev_t st_rdev;
400     off_t st_size;
401     struct timespec st_atim;
402     struct timespec st_mtim;
403     struct timespec st_ctim;
404     blksize_t st_blksize;
405     blkcnt_t st_blocks;
406     unsigned long __unused[3];
407 }
408 ;
409 struct stat64
410 {
411     dev_t st_dev;
412     ino64_t st_ino;
413     nlink_t st_nlink;
414     mode_t st_mode;
415     uid_t st_uid;
416     gid_t st_gid;
417     unsigned int pad0;
418     dev_t st_rdev;
419     off_t st_size;
420     struct timespec st_atim;
421     struct timespec st_mtim;
422     struct timespec st_ctim;
423     blksize_t st_blksize;
424     blkcnt64_t st_blocks;
425     unsigned long __unused[3];
426 }
427 ;

```

1.3.15. sys/statvfs.h

```

428
429 struct statvfs
430 {
431     unsigned long f_bsize;
432     unsigned long f_frsize;
433     fsblkcnt64_t f_blocks;
434     fsblkcnt64_t f_bfree;
435     fsblkcnt64_t f_bavail;
436     fsfilcnt64_t f_files;
437     fsfilcnt64_t f_ffree;
438     fsfilcnt64_t f_favail;
439     unsigned long f_fsid;
440     unsigned long f_flag;
441     unsigned long f_namemax;
442     unsigned int __f_spare[6];
443 }
444 ;
445 struct statvfs64

```

```

446 {
447     unsigned long f_bsize;
448     unsigned long f_frsize;
449     fsblkcnt64_t f_blocks;
450     fsblkcnt64_t f_bfree;
451     fsblkcnt64_t f_bavail;
452     fsfilcnt64_t f_files;
453     fsfilcnt64_t f_ffree;
454     fsfilcnt64_t f_favail;
455     unsigned long f_fsid;
456     unsigned long f_flag;
457     unsigned long f_namemax;
458     unsigned int __f_spare[6];
459 }
460 ;

```

1.3.16. sys/types.h

```

461
462 typedef long int64_t;
463
464 typedef int64_t ssize_t;

```

1.3.17. termios.h

```

465
466 #define OLCUC    0000002
467 #define ONLCR   0000004
468 #define XCASE   0000004
469 #define NLDLY   0000400
470 #define CR1     0001000
471 #define IUCLC   0001000
472 #define CR2     0002000
473 #define CR3     0003000
474 #define CRDLY   0003000
475 #define TAB1    0004000
476 #define TAB2    0010000
477 #define TAB3    0014000
478 #define TABDLY  0014000
479 #define BS1     0020000
480 #define BSDLY   0020000
481 #define VT1     0040000
482 #define VTDLY   0040000
483 #define FF1     0100000
484 #define FFDLY   0100000
485
486 #define VSUSP   10
487 #define VEOL    11
488 #define VREPRINT 12
489 #define VDISCARD 13
490 #define VWERASE 14
491 #define VEOL2   16

```

```

492 #define VMIN      6
493 #define VSWTC     7
494 #define VSTART    8
495 #define VSTOP     9
496
497 #define IXON      0002000
498 #define IXOFF    0010000
499
500 #define CS6       0000020
501 #define CS7       0000040
502 #define CS8       0000060
503 #define CSIZE     0000060
504 #define CSTOPB   0000100
505 #define CREAD     0000200
506 #define PARENB   0000400
507 #define PARODD   0001000
508 #define HUPCL    0002000
509 #define CLOCAL   0004000
510 #define VTIME    5
511
512 #define ISIG      0000001
513 #define ICANON    0000002
514 #define ECHOE     0000020
515 #define ECHOK     0000040
516 #define ECHONL   0000100
517 #define NOFLSH   0000200
518 #define TOSTOP   0000400
519 #define ECHOCTL   0001000
520 #define ECHOPRT  0002000
521 #define ECHOKE    0004000
522 #define FLUSHO   0010000
523 #define PENDIN   0040000
524 #define IEXTEN   0100000

```

1.3.18. ucontext.h

```

525
526 #define _SC_GR0_OFFSET (((char *) & ((struct sigcontext *) 0)->sc_gr[0]) - (char *) 0)
527
528 typedef struct sigcontext mcontext_t;
529
530 typedef struct ucontext
531 {
532     union
533     {
534         mcontext_t _mc;
535         struct
536         {
537             unsigned long _pad[_SC_GR0_OFFSET / 8];
538             struct ucontext *_link;
539         }
540         _uc;

```

```

541     }
542     _u;
543 }
544 ucontext_t;

```

1.3.19. unistd.h

```

545
546 typedef long intptr_t;

```

1.3.20. utmp.h

```

547
548 struct lastlog
549 {
550     int32_t ll_time;
551     char ll_line[UT_LINESIZE];
552     char ll_host[UT_HOSTSIZE];
553 }
554 ;
555
556 struct utmp
557 {
558     short ut_type;
559     pid_t ut_pid;
560     char ut_line[UT_LINESIZE];
561     char ut_id[4];
562     char ut_user[UT_NAMESIZE];
563     char ut_host[UT_HOSTSIZE];
564     struct exit_status ut_exit;
565     long ut_session;
566     struct timeval ut_tv;
567     int32_t ut_addr_v6[4];
568     char __unused[20];
569 }
570 ;

```

1.3.21. utmpx.h

```

571
572 struct utmpx
573 {
574     short ut_type;
575     pid_t ut_pid;
576     char ut_line[UT_LINESIZE];
577     char ut_id[4];
578     char ut_user[UT_NAMESIZE];
579     char ut_host[UT_HOSTSIZE];
580     struct exit_status ut_exit;
581     long ut_session;
582     struct timeval ut_tv;
583     int32_t ut_addr_v6[4];

```

```

584     char __unused[20];
585 }
586 ;

```

1.4. Interfaces for libm

587 Table 1-28 defines the library name and shared object name for the libm library

588 **Table 1-28. libm Definition**

Library:	libm
SONAME:	libm.so.6.1

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

ISO/IEC 9899: C (1999, Programming Languages—C)
 CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,
 C606) SUSv2
 591 ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)

1.4.1. Math

592 1.4.1.1. Interfaces for Math

593 An LSB conforming implementation shall provide the architecture specific functions for Math specified in Table 1-29,
 594 with the full functionality as described in the referenced underlying specification.

595 **Table 1-29. libm - Math Function Interfaces**

<code>aeos(GLIBC_2.2)</code> <code>acos(GLIBC_2.2)</code> [1]	<code>eeexp(GLIBC_2.2)</code> <code>exp(GLIBC_2.2)</code> [1]	<code>expf(GLIBC_2.2)</code> <code>expf(GLIBC_2.2)</code> [1]	<code>jnf(GLIBC_2.2)</code> <code>jnf(GLIBC_2.2)</code> [2]	<code>remquof(GLIBC_2.2)</code> <code>remquof(GLIBC_2.2)</code> [1]
<code>aeosf(GLIBC_2.2)</code> <code>acosf(GLIBC_2.2)</code> [1]	<code>eeexpf(GLIBC_2.2)</code> <code>expf(GLIBC_2.2)</code> [1]	<code>expl(GLIBC_2.2)</code> <code>expl(GLIBC_2.2)</code> [1]	<code>jnl(GLIBC_2.2)</code> <code>jnl(GLIBC_2.2)</code> [2]	<code>remquo(GLIBC_2.2)</code> <code>remquo(GLIBC_2.2)</code> [1]
<code>aeosh(GLIBC_2.2)</code> <code>cosh(GLIBC_2.2)</code> [1]	<code>eeexpl(GLIBC_2.2)</code> <code>expl(GLIBC_2.2)</code> [1]	<code>expm1(GLIBC_2.2)</code> <code>expm1(GLIBC_2.2)</code> [1]	<code>ldexp(GLIBC_2.2)</code> <code>dexp(GLIBC_2.2)</code> [1]	<code>rint(GLIBC_2.2)</code> <code>rint(GLIBC_2.2)</code> [1]
<code>aeoshf(GLIBC_2.2)</code> <code>acoshf(GLIBC_2.2)</code> [1]	<code>eimag(GLIBC_2.2)</code> <code>cimag(GLIBC_2.2)</code> [1]	<code>fabs(GLIBC_2.2)</code> <code>fabs(GLIBC_2.2)</code> [1]	<code>ldexpf(GLIBC_2.2)</code> <code>dexpf(GLIBC_2.2)</code> [1]	<code>rintf(GLIBC_2.2)</code> <code>rintf(GLIBC_2.2)</code> [1]
<code>aeoshl(GLIBC_2.2)</code> <code>acoshl(GLIBC_2.2)</code> [1]	<code>eimagf(GLIBC_2.2)</code> <code>cimagf(GLIBC_2.2)</code> [1]	<code>fabsf(GLIBC_2.2)</code> <code>fabsf(GLIBC_2.2)</code> [1]	<code>ldexpl(GLIBC_2.2)</code> <code>dexpl(GLIBC_2.2)</code> [1]	<code>rintl(GLIBC_2.2)</code> <code>rintl(GLIBC_2.2)</code> [1]
<code>aeosl(GLIBC_2.2)</code> <code>cosl(GLIBC_2.2)</code>	<code>eimagl(GLIBC_2.2)</code> <code>cimagl(GLIBC_2.2)</code>	<code>fabsl(GLIBC_2.2)</code> <code>fabsl(GLIBC_2.2)</code> [1]	<code>lgamma(GLIBC_2.2)</code> <code>lgamma(GLIBC_2.2)</code>	<code>round(GLIBC_2.2)</code> <code>round(GLIBC_2.2)</code>

[1]	[1]		2.2) [1]	[1]
asin(GLIBC_2.2)asin(GLIBC_2.2) [1]	elog(GLIBC_2.2)clog(GLIBC_2.2) [1]	fdim(GLIBC_2.2)fdim(GLIBC_2.2) [1]	lgamma_r(GLIBC_2.2)lgamma_r(GLIBC_2.2) [2]	roundf(GLIBC_2.2)roundf(GLIBC_2.2) [1]
asinf(GLIBC_2.2)asinf(GLIBC_2.2) [1]	elog10(GLIBC_2.2)clog10(GLIBC_2.2) [2]	fdimf(GLIBC_2.2)fdimf(GLIBC_2.2) [1]	lgammaf(GLIBC_2.2)lgammaf(GLIBC_2.2) [1]	roundl(GLIBC_2.2)roundl(GLIBC_2.2) [1]
asinh(GLIBC_2.2)asinh(GLIBC_2.2) [1]	elog10f(GLIBC_2.2)clog10f(GLIBC_2.2) [2]	fdiml(GLIBC_2.2)fdiml(GLIBC_2.2) [1]	lgammaf_r(GLIBC_2.2)lgammaf_r(GLIBC_2.2) [2]	scalb(GLIBC_2.2)scalb(GLIBC_2.2) [1]
asinhf(GLIBC_2.2)asinhf(GLIBC_2.2) [1]	elog10l(GLIBC_2.2)clog10l(GLIBC_2.2) [2]	feclearexcept(GLIBC_2.2)feclearexcept(GLIBC_2.2) [1]	lgammal(GLIBC_2.2)lgammal(GLIBC_2.2) [1]	scalbf(GLIBC_2.2)scalbf(GLIBC_2.2) [2]
asinhl(GLIBC_2.2)asinhl(GLIBC_2.2) [1]	elogf(GLIBC_2.2)clogf(GLIBC_2.2) [1]	fegetenv(GLIBC_2.2)fegetenv(GLIBC_2.2) [1]	lgammal_r(GLIBC_2.2)lgammal_r(GLIBC_2.2) [2]	scalbl(GLIBC_2.2)scalbl(GLIBC_2.2) [2]
asinl(GLIBC_2.2)asinl(GLIBC_2.2) [1]	elogl(GLIBC_2.2)clogl(GLIBC_2.2) [1]	fegetexceptflag(GLIBC_2.2)fegetexceptflag(GLIBC_2.2) [1]	llrint(GLIBC_2.2)llrint(GLIBC_2.2) [1]	scalbln(GLIBC_2.2)scalbln(GLIBC_2.2) [1]
atan(GLIBC_2.2)atan(GLIBC_2.2) [1]	eonj(GLIBC_2.2)conj(GLIBC_2.2) [1]	fegetround(GLIBC_2.2)fegetround(GLIBC_2.2) [1]	llrintf(GLIBC_2.2)llrintf(GLIBC_2.2) [1]	scalblnf(GLIBC_2.2)scalblnf(GLIBC_2.2) [1]
atan2(GLIBC_2.2)atan2(GLIBC_2.2) [1]	eonjf(GLIBC_2.2)conjf(GLIBC_2.2) [1]	feholdexcept(GLIBC_2.2)feholdexcept(GLIBC_2.2) [1]	llrintl(GLIBC_2.2)llrintl(GLIBC_2.2) [1]	scalblnl(GLIBC_2.2)scalblnl(GLIBC_2.2) [1]
atan2f(GLIBC_2.2)atan2f(GLIBC_2.2) [1]	eonjlf(GLIBC_2.2)conjlf(GLIBC_2.2) [1]	feraiseexcept(GLIBC_2.2)feraiseexcept(GLIBC_2.2) [1]	llround(GLIBC_2.2)llround(GLIBC_2.2) [1]	scalbn(GLIBC_2.2)scalbn(GLIBC_2.2) [1]
atan2l(GLIBC_2.2)atan2l(GLIBC_2.2) [1]	eopysign(GLIBC_2.2)copysign(GLIBC_2.2) [1]	fesetenv(GLIBC_2.2)fesetenv(GLIBC_2.2) [1]	llroundf(GLIBC_2.2)llroundf(GLIBC_2.2) [1]	scalbnf(GLIBC_2.2)scalbnf(GLIBC_2.2) [1]
atanf(GLIBC_2.2)atanf(GLIBC_2.2) [1]	eopysignf(GLIBC_2.2)copysignf(GLIBC_2.2) [1]	fesetexceptflag(GLIBC_2.2)fesetexceptflag(GLIBC_2.2) [1]	llroundl(GLIBC_2.2)llroundl(GLIBC_2.2) [1]	scalbnl(GLIBC_2.2)scalbnl(GLIBC_2.2) [1]
atanh(GLIBC_2.2)atanh(GLIBC_2.2) [1]	eopysignl(GLIBC_2.2)copysignl(GLIBC_2.2) [1]	fesetround(GLIBC_2.2)fesetround(GLIBC_2.2) [1]	log(GLIBC_2.2)log(GLIBC_2.2) [1]	significand(GLIBC_2.2)significand(GLIBC_2.2) [2]
atanhf(GLIBC_2.2)atanhf(GLIBC_2.2) [1]	eos(GLIBC_2.2)cos(GLIBC_2.2) [1]	fetestexcept(GLIBC_2.2)fetestexcept(G	log10(GLIBC_2.2)log10(GLIBC_2.2)	significandf(GLIBC_2.2)significandf(G

[1]		LIBC_2.2) [1]	[1]	LIBC_2.2) [2]
atanhl(GLIBC_2.2) atanhl(GLIBC_2.2) [1]	eosf(GLIBC_2.2)co sf(GLIBC_2.2) [1]	feupdateenv(GLIBC _2.2)feupdateenv(G LIBC_2.2) [1]	log10f(GLIBC_2.2) log10f(GLIBC_2.2) [1]	significandl(GLIBC _2.2)significandl(G LIBC_2.2) [2]
atanl(GLIBC_2.2)at anl(GLIBC_2.2) [1]	eosh(GLIBC_2.2)co sh(GLIBC_2.2) [1]	finite(GLIBC_2.2)fi nite(GLIBC_2.2) [3]	log10l(GLIBC_2.2)l og10l(GLIBC_2.2) [1]	sin(GLIBC_2.2)sin(GLIBC_2.2) [1]
eabs(GLIBC_2.2)ca bs(GLIBC_2.2) [1]	eoshf(GLIBC_2.2)c oshf(GLIBC_2.2) [1]	finitef(GLIBC_2.2)f initf(GLIBC_2.2) [2]	log1pf(GLIBC_2.2)l og1p(GLIBC_2.2) [1]	sineos(GLIBC_2.2) sincos(GLIBC_2.2) [2]
eabsf(GLIBC_2.2)c absf(GLIBC_2.2) [1]	eoshl(GLIBC_2.2)c oshl(GLIBC_2.2) [1]	finitel(GLIBC_2.2)f initel(GLIBC_2.2) [2]	logb(GLIBC_2.2)lo gb(GLIBC_2.2) [1]	sineosf(GLIBC_2.2) sincosf(GLIBC_2.2) [2]
eabsl(GLIBC_2.2)c absl(GLIBC_2.2) [1]	eosl(GLIBC_2.2)co sl(GLIBC_2.2) [1]	floor(GLIBC_2.2)fl oor(GLIBC_2.2) [1]	logf(GLIBC_2.2)lo gf(GLIBC_2.2) [1]	sineosl(GLIBC_2.2) sincosl(GLIBC_2.2) [2]
eacos(GLIBC_2.2)c acos(GLIBC_2.2) [1]	epow(GLIBC_2.2)c pow(GLIBC_2.2) [1]	floorf(GLIBC_2.2)f loorf(GLIBC_2.2) [1]	logl(GLIBC_2.2)log l(GLIBC_2.2) [1]	sinf(GLIBC_2.2)sin f(GLIBC_2.2) [1]
eacosf(GLIBC_2.2) cacosf(GLIBC_2.2) [1]	epowf(GLIBC_2.2) cpowf(GLIBC_2.2) [1]	floorl(GLIBC_2.2)fl oorl(GLIBC_2.2) [1]	rint(GLIBC_2.2)lri nt(GLIBC_2.2) [1]	sinh(GLIBC_2.2)sin h(GLIBC_2.2) [1]
eacosh(GLIBC_2.2) cacosh(GLIBC_2.2) [1]	epowl(GLIBC_2.2) cpowl(GLIBC_2.2) [1]	fma(GLIBC_2.2)fm a(GLIBC_2.2) [1]	rintf(GLIBC_2.2)lr intf(GLIBC_2.2) [1]	sinhf(GLIBC_2.2)si nhf(GLIBC_2.2) [1]
eacoshf(GLIBC_2.2) cacoshf(GLIBC_2. 2) [1]	eprojf(GLIBC_2.2)c projf(GLIBC_2.2) [1]	fmaf(GLIBC_2.2)f maf(GLIBC_2.2) [1]	rintl(GLIBC_2.2)lri ntl(GLIBC_2.2) [1]	sinhl(GLIBC_2.2)si nhl(GLIBC_2.2) [1]
eacoshl(GLIBC_2.2) cacoshl(GLIBC_2. 2) [1]	eprojf(GLIBC_2.2)c projf(GLIBC_2.2) [1]	fmal(GLIBC_2.2)f mal(GLIBC_2.2) [1]	lround(GLIBC_2.2) lround(GLIBC_2.2) [1]	sinl(GLIBC_2.2)sin l(GLIBC_2.2) [1]
eacosl(GLIBC_2.2) cacosl(GLIBC_2.2) [1]	eprojl(GLIBC_2.2)c projl(GLIBC_2.2) [1]	fmax(GLIBC_2.2)f max(GLIBC_2.2) [1]	lroundf(GLIBC_2.2) lroundf(GLIBC_2. 2) [1]	sqrt(GLIBC_2.2)sqr t(GLIBC_2.2) [1]
earg(GLIBC_2.2)ca rg(GLIBC_2.2) [1]	erealf(GLIBC_2.2)cr eal(GLIBC_2.2) [1]	fmaxf(GLIBC_2.2)f maxf(GLIBC_2.2) [1]	lroundl(GLIBC_2.2) lroundl(GLIBC_2. 2) [1]	sqrtf(GLIBC_2.2)sq rtf(GLIBC_2.2) [1]
eargf(GLIBC_2.2)c argf(GLIBC_2.2) [1]	erealf(GLIBC_2.2)c realf(GLIBC_2.2) [1]	fmaxl(GLIBC_2.2)f maxl(GLIBC_2.2) [1]	matherr(GLIBC_2.2) matherr(GLIBC_2. 2) [2]	sqrtl(GLIBC_2.2)sq rtl(GLIBC_2.2) [1]

t(GLIBC_2.2) [1]	ctanhf(GLIBC_2.2) [1]	hypotf(GLIBC_2.2) [1]	nexttowardf(GLIBC_2.2) [1]	(GLIBC_2.2) [2]
ertrf(GLIBC_2.2) [1]	etanhf(GLIBC_2.2) [1]	hypotl(GLIBC_2.2) [1]	nexttowardl(GLIBC_2.2) [1]	y0l(GLIBC_2.2)y0l (GLIBC_2.2) [2]
ertrl(GLIBC_2.2) [1]	etanl(GLIBC_2.2) [1]	ilogb(GLIBC_2.2) [1]	pow(GLIBC_2.2) [1]	y1(GLIBC_2.2)y1 (GLIBC_2.2) [1]
eeos(GLIBC_2.2) [1]	dremf(GLIBC_2.2) [2]	ilogbf(GLIBC_2.2) [1]	pow10(GLIBC_2.2) [2]	y1f(GLIBC_2.2)y1f (GLIBC_2.2) [2]
eeosf(GLIBC_2.2) [1]	dremf(GLIBC_2.2) [2]	ilogbl(GLIBC_2.2) [1]	pow10f(GLIBC_2.2) [2]	y1l(GLIBC_2.2)y1l (GLIBC_2.2) [2]
eeosh(GLIBC_2.2) [1]	erf(GLIBC_2.2) [1]	j0(GLIBC_2.2) [1]	pow10l(GLIBC_2.2) [2]	yn(GLIBC_2.2)yn (GLIBC_2.2) [1]
eeoshf(GLIBC_2.2) [1]	erfc(GLIBC_2.2) [1]	j0f(GLIBC_2.2) [2]	powf(GLIBC_2.2) [1]	ynf(GLIBC_2.2)ynf (GLIBC_2.2) [2]
eeoshl(GLIBC_2.2) [1]	erfcf(GLIBC_2.2) [1]	j0l(GLIBC_2.2) [2]	powl(GLIBC_2.2) [1]	ynl(GLIBC_2.2)ynl (GLIBC_2.2) [2]
eeosl(GLIBC_2.2) [1]	erfel(GLIBC_2.2) [1]	j1(GLIBC_2.2) [1]	remainder(GLIBC_2.2) [1]	
eeil(GLIBC_2.2) [1]	erff(GLIBC_2.2) [1]	j1f(GLIBC_2.2) [2]	remainderf(GLIBC_2.2) [1]	
eeilf(GLIBC_2.2) [1]	erfl(GLIBC_2.2) [1]	j1l(GLIBC_2.2) [2]	remainderl(GLIBC_2.2) [1]	
eeill(GLIBC_2.2) [1]	exp(GLIBC_2.2) [1]	jnl(GLIBC_2.2) [1]	remquo(GLIBC_2.2) [1]	

596

597 *Referenced Specification(s)*598 **[1].** ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)
599 V3)600 **[2].** ISO/IEC 9899: C (1999, Programming Language—C)

601 ~~[3]. CAE Specification, January 1997, System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0,~~
 602 ~~€606)SUSv2~~

603 An LSB conforming implementation shall provide the architecture specific data interfaces for Math specified in Table
 604 1-30, with the full functionality as described in the referenced underlying specification.

605 **Table 1-30. libm - Math Data Interfaces**

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

607 *Referenced Specification(s)*

608 ~~[1]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)~~
 609 ~~V3)~~

1.5. Interfaces for libpthread

610 Table 1-31 defines the library name and shared object name for the libpthread library

611 **Table 1-31. libpthread Definition**

612	Library:	libpthread
	SONAME:	libpthread.so.0

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

Large File Support

~~Linux Standard Base~~this specification

614 ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

1.5.1. Realtime Threads

1.5.1.1. Interfaces for Realtime Threads

616 No external functions are defined for libpthread - Realtime Threads

1.5.2. Advanced Realtime Threads

1.5.2.1. Interfaces for Advanced Realtime Threads

618 No external functions are defined for libpthread - Advanced Realtime Threads

1.5.3. Posix Threads

1.5.3.1. Interfaces for Posix Threads

620 An LSB conforming implementation shall provide the architecture specific functions for Posix Threads specified in
 621 Table 1-32, with the full functionality as described in the referenced underlying specification.

622 **Table 1-32. libpthread - Posix Threads Function Interfaces**

<code>_pthread_cleanup_pop(GLIBC_2.2)_pthread_cleanup_pop(GLIBC_2.2) [1]</code>	<code>pthread_cancel(GLIBC_2.2)pthread_cancel(GLIBC_2.2) [2]</code>	<code>pthread_join(GLIBC_2.2)pthread_join(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_destroy(GLIBC_2.2)pthread_rwlock_destroy(GLIBC_2.2) [2]</code>	<code>pthread_setconcurrency(GLIBC_2.2)pthread_setconcurrency(GLIBC_2.2) [2]</code>
<code>_pthread_cleanup_push(GLIBC_2.2)_pthread_cleanup_push(GLIBC_2.2) [1]</code>	<code>pthread_cond_broadcast(GLIBC_2.3.2)pthread_cond_broadcast(GLIBC_2.3.2) [2]</code>	<code>pthread_key_create(GLIBC_2.2)pthread_key_create(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_init(GLIBC_2.2)pthread_rwlock_init(GLIBC_2.2) [2]</code>	<code>pthread_setspecific(GLIBC_2.2)pthread_setspecific(GLIBC_2.2) [2]</code>
<code>pread(GLIBC_2.2)pread(GLIBC_2.2) [2]</code>	<code>pthread_cond_destroy(GLIBC_2.3.2)pthread_cond_destroy(GLIBC_2.3.2) [2]</code>	<code>pthread_key_delete(GLIBC_2.2)pthread_key_delete(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_rdlock(GLIBC_2.2)pthread_rwlock_rdlock(GLIBC_2.2) [2]</code>	<code>pthread_sigmask(GLIBC_2.2)pthread_sigmask(GLIBC_2.2) [2]</code>
<code>pread64(GLIBC_2.2)pread64(GLIBC_2.2) [3]</code>	<code>pthread_cond_init(GLIBC_2.3.2)pthread_cond_init(GLIBC_2.3.2) [2]</code>	<code>pthread_kill(GLIBC_2.2)pthread_kill(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_timedrdlock(GLIBC_2.2)pthread_rwlock_timedrdlock(GLIBC_2.2) [2]</code>	<code>pthread_testcancel(GLIBC_2.2)pthread_testcancel(GLIBC_2.2) [2]</code>
<code>pthread_attr_destroy(GLIBC_2.2)pthread_attr_destroy(GLIBC_2.2) [2]</code>	<code>pthread_cond_signal(GLIBC_2.3.2)pthread_cond_signal(GLIBC_2.3.2) [2]</code>	<code>pthread_mutex_destroy(GLIBC_2.2)pthread_mutex_destroy(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_timedwrlock(GLIBC_2.2)pthread_rwlock_timedwrlock(GLIBC_2.2) [2]</code>	<code>pwrite(GLIBC_2.2)pwrite(GLIBC_2.2) [2]</code>
<code>pthread_attr_getdetachstate(GLIBC_2.2)pthread_attr_getdetachstate(GLIBC_2.2) [2]</code>	<code>pthread_cond_timedwait(GLIBC_2.3.2)pthread_cond_timedwait(GLIBC_2.3.2) [2]</code>	<code>pthread_mutex_init(GLIBC_2.2)pthread_mutex_init(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_tryrdlock(GLIBC_2.2)pthread_rwlock_tryrdlock(GLIBC_2.2) [2]</code>	<code>pwrite64(GLIBC_2.2)pwrite64(GLIBC_2.2) [3]</code>
<code>pthread_attr_getguardsize(GLIBC_2.2)pthread_attr_getguardsize(GLIBC_2.2) [2]</code>	<code>pthread_cond_wait(GLIBC_2.3.2)pthread_cond_wait(GLIBC_2.3.2) [2]</code>	<code>pthread_mutex_lock(GLIBC_2.2)pthread_mutex_lock(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_trywrlock(GLIBC_2.2)pthread_rwlock_trywrlock(GLIBC_2.2) [2]</code>	<code>sem_close(GLIBC_2.2)sem_close(GLIBC_2.2) [2]</code>
<code>pthread_attr_getschedparam(GLIBC_2.2)pthread_attr_getschedparam(GLIBC_2.2) [2]</code>	<code>pthread_condattr_destroy(GLIBC_2.2)pthread_condattr_destroy(GLIBC_2.2) [2]</code>	<code>pthread_mutex_trylock(GLIBC_2.2)pthread_mutex_trylock(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_unlock(GLIBC_2.2)pthread_rwlock_unlock(GLIBC_2.2) [2]</code>	<code>sem_destroy(GLIBC_2.2)sem_destroy(GLIBC_2.2) [2]</code>
<code>pthread_attr_getstackaddr(GLIBC_2.2)pthread_attr_getstackaddr(GLIBC_2.2) [2]</code>	<code>pthread_condattr_getshared(GLIBC_2.2)pthread_condattr_getshared(GLIBC_2.2) [2]</code>	<code>pthread_mutex_unlock(GLIBC_2.2)pthread_mutex_unlock(GLIBC_2.2) [2]</code>	<code>pthread_rwlock_wrl</code>	<code>sem_getvalue(GLIBC_2.2)sem_getvalue(GLIBC_2.2) [2]</code>

addr(GLIBC_2.2) [2]	getpshared(GLIBC_2.2) [2]	(GLIBC_2.2) [2]	k(GLIBC_2.2) [2]	
pthread_attr_getstacksize(GLIBC_2.2)pthread_attr_getstacksize(GLIBC_2.2) [2]	pthread_condattr_init(GLIBC_2.2)pthread_condattr_init(GLIBC_2.2) [2]	pthread_mutexattr_destroy(GLIBC_2.2)pthread_mutexattr_destroy(GLIBC_2.2) [2]	pthread_rwlockattr_destroy(GLIBC_2.2)pthread_rwlockattr_destroy(GLIBC_2.2) [2]	sem_init(GLIBC_2.2)sem_init(GLIBC_2.2) [2]
pthread_attr_init(GLIBC_2.2)pthread_attr_init(GLIBC_2.2) [2]	pthread_condattr_setpshared(GLIBC_2.2)pthread_condattr_setpshared(GLIBC_2.2) [2]	pthread_mutexattr_getpshared(GLIBC_2.2)pthread_mutexattr_getpshared(GLIBC_2.2) [2]	pthread_rwlockattr_getpshared(GLIBC_2.2)pthread_rwlockattr_getpshared(GLIBC_2.2) [2]	sem_open(GLIBC_2.2)sem_open(GLIBC_2.2) [2]
pthread_attr_setdetachstate(GLIBC_2.2)pthread_attr_setdetachstate(GLIBC_2.2) [2]	pthread_create(GLIBC_2.2)pthread_create(GLIBC_2.2) [2]	pthread_mutexattr_gettype(GLIBC_2.2)pthread_mutexattr_gettype(GLIBC_2.2) [2]	pthread_rwlockattr_init(GLIBC_2.2)pthread_rwlockattr_init(GLIBC_2.2) [2]	sem_post(GLIBC_2.2)sem_post(GLIBC_2.2) [2]
pthread_attr_setguardsize(GLIBC_2.2)pthread_attr_setguardsize(GLIBC_2.2) [2]	pthread_detach(GLIBC_2.2)pthread_detach(GLIBC_2.2) [2]	pthread_mutexattr_init(GLIBC_2.2)pthread_mutexattr_init(GLIBC_2.2) [2]	pthread_rwlockattr_setpshared(GLIBC_2.2)pthread_rwlockattr_setpshared(GLIBC_2.2) [2]	sem_timedwait(GLIBC_2.2)sem_timedwait(GLIBC_2.2) [2]
pthread_attr_setschedparam(GLIBC_2.2)pthread_attr_setschedparam(GLIBC_2.2) [2]	pthread_equal(GLIBC_2.2)pthread_equal(GLIBC_2.2) [2]	pthread_mutexattr_setpshared(GLIBC_2.2)pthread_mutexattr_setpshared(GLIBC_2.2) [2]	pthread_self(GLIBC_2.2)pthread_self(GLIBC_2.2) [2]	sem_trywait(GLIBC_2.2)sem_trywait(GLIBC_2.2) [2]
pthread_attr_setstackaddr(GLIBC_2.2)pthread_attr_setstackaddr(GLIBC_2.2) [2]	pthread_exit(GLIBC_2.2)pthread_exit(GLIBC_2.2) [2]	pthread_mutexattr_settype(GLIBC_2.2)pthread_mutexattr_settype(GLIBC_2.2) [2]	pthread_setcancelstate(GLIBC_2.2)pthread_setcancelstate(GLIBC_2.2) [2]	sem_unlink(GLIBC_2.2)sem_unlink(GLIBC_2.2) [2]
pthread_attr_setstacksize(GLIBC_2.3.3)pthread_attr_setstacksize(GLIBC_2.3.3) [2]	pthread_getspecific(GLIBC_2.2)pthread_getspecific(GLIBC_2.2) [2]	pthread_once(GLIBC_2.2)pthread_once(GLIBC_2.2) [2]	pthread_setcanceltype(GLIBC_2.2)pthread_setcanceltype(GLIBC_2.2) [2]	sem_wait(GLIBC_2.2)sem_wait(GLIBC_2.2) [2]

623

624 *Referenced Specification(s)*

625 [1]. Linux Standard Basethis specification

626 [2]. ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS)

627 V3)

628 [3]. Large File Support

1.6. Interfaces for libgcc_s

629 Table 1-33 defines the library name and shared object name for the libgcc_s library

630 **Table 1-33. libgcc_s Definition**

Library:	libgcc_s
SONAME:	libgcc_s.so.1

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

633 ~~Linux Standard Base~~this specification

1.6.1. Unwind Library

1.6.1.1. Interfaces for Unwind Library

634 An LSB conforming implementation shall provide the architecture specific functions for Unwind Library specified in
635 Table 1-34, with the full functionality as described in the referenced underlying specification.

637 **Table 1-34. libgcc_s - Unwind Library Function Interfaces**

_Unwind_DeleteException(GCC_3.0) _Unwind_DeleteException(GCC_3.0) [1]	_Unwind_GetGR(GCC_3.0) _Unwind_GetGR(GCC_3.0) [1]	_Unwind_GetLanguageSpecificData(GCC_3.0) _Unwind_GetLanguageSpecificData(GCC_3.0) [1]	_Unwind_RaiseException(GCC_3.0) _Unwind_RaiseException(GCC_3.0) [1]	_Unwind_SetGR(GCC_3.0) _Unwind_SetGR(GCC_3.0) [1]
_Unwind_ForceUnwind(GCC_3.0) _Unwind_ForceUnwind(GCC_3.0) [1]	_Unwind_GetIP(GCC_3.0) _Unwind_GetIP(GCC_3.0) [1]	_Unwind_GetRegionStart(GCC_3.0) _Unwind_GetRegionStart(GCC_3.0) [1]	_Unwind_Resume(GCC_3.0) _Unwind_Resume(GCC_3.0) [1]	_Unwind_SetIP(GCC_3.0) _Unwind_SetIP(GCC_3.0) [1]

639 *Referenced Specification(s)*

640 [1]. ~~Linux Standard Base~~this specification

1.7. Interface Definitions for libgcc_s

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

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

`_Unwind_DeleteException`

Name

644 `_Unwind_DeleteException` — private C++ error handling method

Synopsis

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

Description

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

`_Unwind_ForcedUnwind`

Name

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

Synopsis

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

Description

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

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

Return Value

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

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

665 `_URC_NO_REASON`

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

669 `_URC_END_OF_STACK`

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

673 `_URC_FATAL_PHASE2_ERROR`

674 The *stop* function may return this code for other fatal conditions like stack corruption.

`_Unwind_GetGR`

Name

675 `_Unwind_GetGR` — private C++ error handling method

Synopsis

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

Description

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

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

`_Unwind_GetIP`

Name

681 `_Unwind_GetIP` — private C++ error handling method

Synopsis

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

Description

683 `_Unwind_GetIP` returns the instruction pointer value for the routine identified by the unwind *context*.

`_Unwind_GetLanguageSpecificData`

Name

684 `_Unwind_GetLanguageSpecificData` — private C++ error handling method

Synopsis

685 `_Unwind_Ptr _Unwind_GetLanguageSpecificData((struct _Unwind_Context *context), uint`
686 `value);`

Description

687 `_Unwind_GetLanguageSpecificData` returns the address of the language specific data area for the current stack
688 frame.

`_Unwind_GetRegionStart`

Name

689 `_Unwind_GetRegionStart` — private C++ error handling method

Synopsis

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

Description

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

`_Unwind_RaiseException`

Name

693 `_Unwind_RaiseException` — private C++ error handling method

Synopsis

694 `_Unwind_Reason_Code _Unwind_RaiseException((struct _Unwind_Exception *object));`

Description

695 `_Unwind_RaiseException` raises an exception, passing along the given exception *object*, which should have its
 696 *exception_class* and *exception_cleanup* fields set. The exception object has been allocated by the
 697 language-specific runtime, and has a language-specific format, exception that it shall contain an
 698 `_Unwind_Exception`.

Return Value

699 `_Unwind_RaiseException` does not return unless an error condition is found. If an error condition occurs, an
 700 `_Unwind_Reason_Code` is returned:

701 `_URC_END_OF_STACK`

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

704 `_URC_FATAL_PHASE1_ERROR`

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

708 `_URC_FATAL_PHASE2_ERROR`

709 The unwinder encountered an unexpected error during phase two. This is usually a *throw*, which will call
 710 `terminate`.

`_Unwind_Resume`

Name

711 `_Unwind_Resume` — private C++ error handling method

Synopsis

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

Description

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

`_Unwind_SetGR`

Name

715 `_Unwind_SetGR` — private C++ error handling method

Synopsis

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

Description

717 `_Unwind_SetGR` sets the *value* of the register *indexed* for the routine identified by the unwind *context*.

`_Unwind_SetIP`

Name

718 `_Unwind_SetIP` — private C++ error handling method

Synopsis

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

Description

720 `_Unwind_SetIP` sets the *value* of the instruction pointer for the routine identified by the unwind *context*

1.8. Interfaces for libdl

721 Table 1-35 defines the library name and shared object name for the libdl library

722 **Table 1-35. libdl Definition**

Library:	libdl
SONAME:	libdl.so.2

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

725 ~~Linux Standard Base~~this specification
~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

1.8.1. Dynamic Loader

726 1.8.1.1. Interfaces for Dynamic Loader

727 An LSB conforming implementation shall provide the architecture specific functions for Dynamic Loader specified in
 728 Table 1-36, with the full functionality as described in the referenced underlying specification.

729 **Table 1-36. libdl - Dynamic Loader Function Interfaces**

dldaddr(GLIBC_2.0) dldaddr(GLIBC_2.0) [1]	dldclose(GLIBC_2.0) dldclose(GLIBC_2.0) [2]	dlderror(GLIBC_2.0) dlderror(GLIBC_2.0) [2]	dldopen(GLIBC_2.1) dldopen(GLIBC_2.1) [1]	dldsym(GLIBC_2.0) dldsym(GLIBC_2.0) [1]
------------------------------------------------------------	--------------------------------------------------------------	--------------------------------------------------------------	------------------------------------------------------------	----------------------------------------------------------

731 *Referenced Specification(s)*

732 [1]. ~~Linux Standard Base~~this specification

733 [2]. ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~
 734 ~~V3)~~

1.9. Interfaces for libcrypt

735 Table 1-37 defines the library name and shared object name for the libcrypt library

736 **Table 1-37. libcrypt Definition**

Library:	libcrypt
SONAME:	libcrypt.so.1

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

739 ~~ISO/IEC 9945: POSIX (2003 Portable Operating System (POSIX) and The Single UNIX® Specification (SUS) V3)~~

1.9.1. Encryption

740 1.9.1.1. Interfaces for Encryption

741 An LSB conforming implementation shall provide the architecture specific functions for Encryption specified in Table
 742 1-38, with the full functionality as described in the referenced underlying specification.

743 **Table 1-38. libcrypt - Encryption Function Interfaces**

744	crypt(GLIBC_2.0)cr ypt(GLIBC_2.0) [1]	encrypt(GLIBC_2.0)encrypt(GLIBC_2. 0) [1]	setkey(GLIBC_2.0) setkey(GLIBC_2.0) [1]		
-----	------------------------------------------	--------------------------------------------------	-----------------------------------------------	--	--

745 *Referenced Specification(s)*

746 **[1]. ISO/IEC 9945: POSIX (2003-Portable Operating System(POSIX)and The Single UNIX® Specification(SUS)**
747 **↯3)**

II. Utility Libraries

Chapter 2. Libraries

The Utility libraries are those that are commonly used, but not part of the Single Unix Specification.

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.

2.1. Interfaces for libz

Table 2-1 defines the library name and shared object name for the libz library

Table 2-1. libz Definition

Library:	libz
SONAME:	libz.so.1

2.1.1. Compression Library

2.1.1.1. Interfaces for Compression Library

No external functions are defined for libz - Compression Library

2.2. Interfaces for libncurses

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

Table 2-2. libncurses Definition

Library:	libncurses
SONAME:	libncurses.so.5

2.2.1. Curses

2.2.1.1. Interfaces for Curses

No external functions are defined for libncurses - Curses

2.3. Interfaces for libutil

Table 2-3 defines the library name and shared object name for the libutil library

Table 2-3. libutil Definition

Library:	libutil
----------	---------

SONAME:	libutil.so.1
---------	--------------

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

~~Linux Standard Base~~this specification

2.3.1. Utility Functions

2.3.1.1. Interfaces for Utility Functions

An LSB conforming implementation shall provide the architecture specific functions for Utility Functions specified in Table 2-4, with the full functionality as described in the referenced underlying specification.

Table 2-4. libutil - Utility Functions Function Interfaces

forkpty(GLIBC_2.0) forkpty(GLIBC_2.0) [1]	login_tty(GLIBC_2.0) login_tty(GLIBC_2.0) [1]	logtmp(GLIBC_2.0) logtmp(GLIBC_2.0) [1]		
login(GLIBC_2.0) login(GLIBC_2.0) [1]	logout(GLIBC_2.0) logout(GLIBC_2.0) [1]	openpty(GLIBC_2.0) openpty(GLIBC_2.0) [1]		

Referenced Specification(s)

~~[1]. Linux Standard Base~~this specification

Appendix A. Alphabetical Listing of Interfaces

A.1. libgcc_s

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

2 | ~~Linux Standard Base~~this specification

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

4	_Unwind_DeleteException [1]	_Unwind_GetLanguageSpecificData [1]	_Unwind_SetGR _Unwind_SetGR[1]
	_Unwind_Force Unwind _Unwind_ForceUnwind[1]	_Unwind_GetRegionStart [1]	_Unwind_SetIP _Unwind_SetIP[1]
	_Unwind_GetGR _Unwind_GetGR[1]	_Unwind_RaiseException [1]	
	_Unwind_GetIP _Unwind_GetIP[1]	_Unwind_Resume _Unwind_Resume[1]	

Linux Packaging Specification

2

3 **Linux Packaging Specification**

Table of Contents

I. Package Format and Installation	55
1. Software Installation	1
1.1. Package Dependencies.....	1
1.2. Package Architecture Considerations	1

I. Package Format and Installation

Chapter 1. Software Installation

1.1. Package Dependencies

- 1 The LSB runtime environment shall provide the following dependencies.
- 2 `lsb-core-ia64`
 - 3 This dependency is used to indicate that the application is dependent on features contained in the LSB-Core
 - 4 specification.
- 5 Other LSB modules may add additional dependencies; such dependencies shall have the format `lsb-module-ia64`.

1.2. Package Architecture Considerations

- 6 All packages must specify an architecture of IA64. A LSB runtime environment must accept an architecture of IA64
- 7 even if the native architecture is different.
- 8 The `archnum` value in the Lead Section shall be 0x0009.

Free Documentation License

2 |
3 | **Free Documentation License**

Table of Contents

A. GNU Free Documentation License	1
A.1. PREAMBLE.....	1
A.2. APPLICABILITY AND DEFINITIONS.....	1
A.3. VERBATIM COPYING.....	2
A.4. COPYING IN QUANTITY.....	2
A.5. MODIFICATIONS.....	3
A.6. COMBINING DOCUMENTS.....	4
A.7. COLLECTIONS OF DOCUMENTS.....	4
A.8. AGGREGATION WITH INDEPENDENT WORKS.....	4
A.9. TRANSLATION.....	5
A.10. TERMINATION.....	5
A.11. FUTURE REVISIONS OF THIS LICENSE.....	5
A.12. How to use this License for your documents.....	5

Appendix A. 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.

A.1. PREAMBLE

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

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

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

A.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

35 designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not
36 "Transparent" is called "Opaque".

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

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

A.3. VERBATIM COPYING

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

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

A.4. COPYING IN QUANTITY

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

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

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

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

A.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.

111 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
112 Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of
113 Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already
114 includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are
115 acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the
116 previous publisher that added the old one.

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

A.6. COMBINING DOCUMENTS

119 You may combine the Document with other documents released under this License, under the terms defined in section
120 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the
121 original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

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

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

A.7. COLLECTIONS OF DOCUMENTS

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

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

A.8. AGGREGATION WITH INDEPENDENT WORKS

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

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

A.9. TRANSLATION

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

A.10. TERMINATION

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

A.11. FUTURE REVISIONS OF THIS LICENSE

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

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

A.12. How to use this License for your documents

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

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

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

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