



DPC: 20/30397113 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 17 February 2020

Origin: International

Latest date for receipt of comments: 09 April 2020

Project No. 2019/01697

Responsible committee: IST/5 Programming languages, their environments and system software interfaces

Interested committees:

Title: Draft BS ISO/IEC 23360-4-2 Linux Standard Base (LSB). Part 4-2: Core specification for AMD64 (X86-64) architecture.

Please notify the secretary if you are aware of any keywords that might assist in classifying or identifying the standard or if the content of this standard

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD. THIS DRAFT IS NOT CURRENT BEYOND 09 April 2020This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.Information on the co-operating organizations represented on the committees referenced above may be obtained from <https://standardsdevelopment.bsigroup.com/>.

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/Figure/ Table/Note	Type of comment	Commend (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	

DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 23360-4-2

ISO/IEC JTC 1/SC 22

Secretariat: **ANSI**

Voting begins on:
2020-02-14

Voting terminates on:
2020-05-08

Linux Standard Base (LSB) —

Part 4-2: Core specification for AMD64 (X86-64) architecture

ICS: 35.080

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

This document is circulated as received from the committee secretariat.



Reference number
ISO/IEC DIS 23360-4-2:2020(E)
Documents
AVA Document Sharing Hub
© ISO/IEC 2020



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Linux Standard Base Core Specification for X86-64

LSB Core - X86-64 5.0

Copyright © 2015 Linux Foundation

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 may be copyrighted by the following parties:

- The Regents of the University of California
- Free Software Foundation
- Ian F. Darwin
- Paul Vixie
- BSDI (now Wind River)
- Jean-loup Gailly and Mark Adler
- Massachusetts Institute of Technology
- Apple Inc.
- Easy Software Products
- artofcode LLC
- Till Kamppeter
- Manfred Wassman
- Python Software Foundation

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

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

UNIX is a registered trademark of The Open Group.

LSB is a trademark of the Linux Foundation in the United States and other countries.

AMD is a trademark of Advanced Micro Devices, Inc.

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

PowerPC is a registered trademark and PowerPC Architecture is a trademark of the IBM Corporation.

S/390 is a registered trademark of the IBM Corporation.

OpenGL is a registered trademark of Silicon Graphics, Inc.

PAM documentation is Copyright (C) Andrew G. Morgan 1996-9. All rights reserved. Used under the following conditions:

1. Redistributions of source code must retain the above copyright notice, and the entire permission notice in its entirety, including the disclaimer of warranties.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

Contents

Contents	iv
List of Tables	vi
Foreword	xv
Status of this Document	xvii
Introduction	xviii
I Introductory Elements	1
1 Scope.....	2
1.1 General	2
1.2 Module Specific Scope	2
2 References.....	3
2.1 Normative References	3
2.2 Informative References/Bibliography	5
3 Requirements	8
3.1 Relevant Libraries	8
3.2 LSB Implementation Conformance.....	8
3.3 LSB Application Conformance	9
4 Terms and Definitions.....	11
5 Documentation Conventions	13
II Executable and Linking Format (ELF)	14
6 Introduction	15
7 Low Level System Information	16
7.1 Machine Interface.....	16
7.2 Function Calling Sequence	17
7.3 Operating System Interface.....	18
7.4 Process Initialization	18
7.5 Coding Examples.....	19
7.6 C Stack Frame.....	19
7.7 Debug Information.....	19
8 Object Format	20
8.1 Introduction.....	20
8.2 ELF Header	20
8.3 Sections.....	20
8.4 Symbol Table	21
8.5 Relocation	21
9 Program Loading and Dynamic Linking	22
9.1 Introduction.....	22
9.2 Program Header.....	22
9.3 Program Loading	22
9.4 Dynamic Linking.....	22
III Base Libraries	24
10 Libraries	25
10.1 Program Interpreter/Dynamic Linker	25
10.2 Interfaces for libc	25
10.3 Data Definitions for libc	49
10.4 Interface Definitions for libc	68
10.5 Interfaces for libm	69
10.6 Data Definitions for libm	74
10.7 Interface Definitions for libm	76

10.8 Interfaces for libpthread.....	77
10.9 Data Definitions for libpthread.....	82
10.10 Interfaces for libgcc_s.....	84
10.11 Data Definitions for libgcc_s.....	84
10.12 Interface Definitions for libgcc_s.....	85
10.13 Interfaces for libdl.....	86
10.14 Data Definitions for libdl.....	86
10.15 Interfaces for libcrypt.....	87
10.16 Data Definitions for libcrypt.....	87
IV Utility Libraries.....	89
11 Libraries.....	90
11.1 Interfaces for libz.....	90
11.2 Data Definitions for libz.....	90
11.3 Interfaces for libncurses.....	91
11.4 Data Definitions for libncurses.....	91
11.5 Interfaces for libncursesw.....	91
11.6 Data Definitions for libncursesw.....	92
11.7 Interfaces for libutil.....	93
V Base Libraries.....	94
12 Libraries.....	95
12.1 Interfaces for libstdcxx.....	95
12.2 Interface Definitions for libstdcxx.....	228
VI Package Format and Installation.....	229
13 Software Installation.....	230
13.1 Package Dependencies.....	230
13.2 Package Architecture Considerations.....	230
Annex A Alphabetical Listing of Interfaces by Library.....	231
A.1 libc.....	231
A.2 libcrypt.....	247
A.3 libdl.....	247
A.4 libgcc_s.....	247
A.5 libm.....	248
A.6 libpthread.....	253
A.7 librt.....	256
A.8 libutil.....	257
Annex B GNU Free Documentation License (Informative).....	258
B.1 PREAMBLE.....	258
B.2 APPLICABILITY AND DEFINITIONS.....	258
B.3 VERBATIM COPYING.....	259
B.4 COPYING IN QUANTITY.....	259
B.5 MODIFICATIONS.....	260
B.6 COMBINING DOCUMENTS.....	262
B.7 COLLECTIONS OF DOCUMENTS.....	262
B.8 AGGREGATION WITH INDEPENDENT WORKS.....	262
B.9 TRANSLATION.....	262
B.10 TERMINATION.....	263
B.11 FUTURE REVISIONS OF THIS LICENSE.....	263
B.12 How to use this License for your documents.....	263

List of Tables

Table 2-1 Normative References	3
Table 2-2 Other References	6
Table 3-1 Standard Library Names	8
Table 7-1 Non Conforming Instructions	16
Table 8-1 ELF Special Sections	20
Table 8-2 Additional Special Sections	21
Table 10-1 libc Definition	25
Table 10-2 libc - RPC Function Interfaces	25
Table 10-3 libc - RPC Deprecated Function Interfaces	27
Table 10-4 libc - System Calls Function Interfaces	27
Table 10-5 libc - System Calls Deprecated Function Interfaces	29
Table 10-6 libc - Standard I/O Function Interfaces	29
Table 10-7 libc - Standard I/O Deprecated Function Interfaces	31
Table 10-8 Sub Macro1()	31
Table 10-9 libc - Signal Handling Function Interfaces	35
Table 10-10 libc - Signal Handling Deprecated Function Interfaces	35
Table 10-11 libc - Signal Handling Data Interfaces	35
Table 10-12 libc - Localization Functions Function Interfaces	36
Table 10-13 libc - Localization Functions Data Interfaces	36
Table 10-14 libc - Posix Spawn Option Function Interfaces	36
Table 10-15 libc - Posix Advisory Option Function Interfaces	37
Table 10-16 libc - Socket Interface Function Interfaces	37
Table 10-17 libc - Socket Interface Data Interfaces	38
Table 10-18 libc - Wide Characters Function Interfaces	38
Table 10-19 libc - String Functions Function Interfaces	39
Table 10-20 libc - String Functions Deprecated Function Interfaces	41
Table 10-21 libc - IPC Functions Function Interfaces	41
Table 10-22 libc - Regular Expressions Function Interfaces	41
Table 10-23 libc - Character Type Functions Function Interfaces	41
Table 10-24 libc - Time Manipulation Function Interfaces	42
Table 10-25 libc - Time Manipulation Data Interfaces	42
Table 10-26 libc - Terminal Interface Functions Function Interfaces	43
Table 10-27 libc - System Database Interface Function Interfaces	43
Table 10-28 libc - System Database Interface Deprecated Function Interfaces	44
Table 10-29 libc - Language Support Function Interfaces	44
Table 10-30 libc - Large File Support Function Interfaces	44
Table 10-31 libc - Large File Support Deprecated Function Interfaces	45
Table 10-32 libc - Standard Library Function Interfaces	45
Table 10-33 libc - Standard Library Deprecated Function Interfaces	48
Table 10-34 libc - Standard Library Data Interfaces	48
Table 10-35 libc - GNU Extensions for libc Function Interfaces	49
Table 10-36 libm Definition	69
Table 10-37 libm - Math Function Interfaces	70
Table 10-38 libm - Math Deprecated Function Interfaces	74
Table 10-39 libm - Math Data Interfaces	74
Table 10-40 libpthread Definition	77
Table 10-41 libpthread - Realtime Threads Function Interfaces	77
Table 10-42 libpthread - Advanced Realtime Threads Function Interfaces	78

Table 10-43 libpthread - Posix Threads Function Interfaces	78
Table 10-44 Tableibpthread - Posix Threads Deprecated Function Interfaces	80
Table 10-45 libpthread - Thread aware versions of libc interfaces Function Interfaces	80
Table 10-46 libpthread - GNU Extensions for libpthread Function Interfaces	80
Table 10-47 libpthread - System Calls Function Interfaces.....	81
Table 10-48 libpthread - Standard I/O Function Interfaces	81
Table 10-49 libpthread - Signal Handling Function Interfaces	81
Table 10-50 libpthread - Standard Library Function Interfaces	82
Table 10-51 libpthread - Socket Interface Function Interfaces	82
Table 10-52 libpthread - Terminal Interface Functions Function Interfaces	82
Table 10-53 libgcc_s Definition.....	84
Table 10-54 libgcc_s - Unwind Library Function Interfaces.....	84
Table 10-55 libdl Definition	86
Table 10-56 libdl - Dynamic Loader Function Interfaces	86
Table 10-57 libcrypt Definition	87
Table 10-58 Tableibcrypt - Encryption Function Interfaces	87
Table 11-1 libz Definition	90
Table 11-2 libncurses Definition	91
Table 11-3 libncursesw Definition	91
Table 11-4 libutil Definition	93
Table 11-5 libutil - Utility Functions Function Interfaces	93
Table 12-1 libstdcxx Definition.....	95
Table 12-2 libstdcxx - C++ Runtime Support Function Interfaces	95
Table 12-3 typeinfo for type_info	96
Table 12-4 typeinfo for __cxxabiv1::__enum_type_info	96
Table 12-5 typeinfo for __cxxabiv1::__array_type_info	97
Table 12-6 Primary vtable for __cxxabiv1::__class_type_info	97
Table 12-7 typeinfo for __cxxabiv1::__class_type_info	98
Table 12-8 libstdcxx - Class __cxxabiv1::__class_type_info Function Interfaces	98
Table 12-9 2-1 Normative References 000.....	99
Table 12-10 typeinfo for __cxxabiv1::__pointer_type_info.....	115
Table 12-11 typeinfo for __cxxabiv1::__function_type_info.....	115
Table 12-12 Primary vtable for __cxxabiv1::__si_class_type_info	116
Table 12-13 typeinfo for __cxxabiv1::__si_class_type_info	117
Table 12-14 libstdcxx - Class __cxxabiv1::__si_class_type_info Function Interfaces ..	117
Table 12-15 Primary vtable for __cxxabiv1::__vmi_class_type_info	117
Table 12-16 typeinfo for __cxxabiv1::__vmi_class_type_info	118
Table 12-17 libstdcxx - Class __cxxabiv1::__vmi_class_type_info Function Interfaces	118
Table 12-18 typeinfo for __cxxabiv1::__fundamental_type_info.....	119
Table 12-19 typeinfo for __cxxabiv1::__pointer_to_member_type_info	119
Table 12-20 libstdcxx - Class __gnu_cxx::__pool_alloc_base Function Interfaces	120
Table 12-21 Primary vtable for __gnu_cxx::stdio_sync_filebuf<char, char_traits<char> >	120
Table 12-22 Primary vtable for __gnu_cxx::stdio_sync_filebuf<wchar_t, char_traits<wchar_t> >	122
Table 12-23 typeinfo for exception	123
Table 12-24 typeinfo for bad_typeid	124
Table 12-25 typeinfo for logic_error.....	124

Table 12-26 typeid for range_error.....	124
Table 12-27 typeid for domain_error.....	125
Table 12-28 typeid for length_error.....	125
Table 12-29 typeid for out_of_range.....	125
Table 12-30 typeid for bad_exception.....	126
Table 12-31 typeid for runtime_error.....	126
Table 12-32 Sub Macro1()	127
Table 12-33 typeid for underflow_error.....	130
Table 12-34 typeid for invalid_argument.....	131
Table 12-35 typeid for bad_cast.....	131
Table 12-36 typeid for bad_alloc.....	131
Table 12-37 typeid for ctype_base.....	134
Table 12-38 libstdcxx - Class ctype<char> Function Interfaces.....	135
Table 12-39 typeid for ctype<wchar_t>.....	135
Table 12-40 libstdcxx - Class ctype<wchar_t> Function Interfaces.....	135
Table 12-41 typeid for ctype_byname<char>.....	136
Table 12-42 libstdcxx - Class ctype_byname<char> Function Interfaces.....	136
Table 12-43 typeid for ctype_byname<wchar_t>.....	136
Table 12-44 libstdcxx - Class ctype_byname<wchar_t> Function Interfaces.....	137
Table 12-45 libstdcxx - Class basic_string<char, char_traits<char>, allocator<char>> Function Interfaces.....	137
Table 12-46 libstdcxx - Class basic_string<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> Function Interfaces.....	142
Table 12-47 Primary vtable for basic_stringstream<char, char_traits<char>, allocator<char>>.....	147
Table 12-48 Secondary vtable for basic_stringstream<char, char_traits<char>, allocator<char>>.....	148
Table 12-49 Secondary vtable for basic_stringstream<char, char_traits<char>, allocator<char>>.....	148
Table 12-50 VTT for basic_stringstream<char, char_traits<char>, allocator<char>>	148
Table 12-51 libstdcxx - Class basic_stringstream<char, char_traits<char>, allocator<char>> Function Interfaces.....	149
Table 12-52 Primary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>.....	149
Table 12-53 Secondary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>.....	149
Table 12-54 Secondary vtable for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>.....	150
Table 12-55 VTT for basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>>.....	150
Table 12-56 libstdcxx - Class basic_stringstream<wchar_t, char_traits<wchar_t>, allocator<wchar_t>> Function Interfaces.....	151
Table 12-57 Primary vtable for basic_istreamstream<char, char_traits<char>, allocator<char>>.....	151
Table 12-58 Secondary vtable for basic_istreamstream<char, char_traits<char>, allocator<char>>.....	151
Table 12-59 VTT for basic_istreamstream<char, char_traits<char>, allocator<char>>	152
Table 12-60 libstdcxx - Class basic_istreamstream<char, char_traits<char>, allocator<char>>.....	152

allocator<char> > Function Interfaces.....	152
Table 12-61 Primary vtable for basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	153
Table 12-62 Secondary vtable for basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	153
Table 12-63 VTT for basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	153
Table 12-64 libstdc++ - Class basic_istream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces	154
Table 12-65 Primary vtable for basic_ostream<char, char_traits<char>, allocator<char> >	154
Table 12-66 Secondary vtable for basic_ostream<char, char_traits<char>, allocator<char> >	154
Table 12-67 VTT for basic_ostream<char, char_traits<char>, allocator<char> >	155
Table 12-68 libstdc++ - Class basic_ostream<char, char_traits<char>, allocator<char> > Function Interfaces.....	155
Table 12-69 Primary vtable for basic_ostream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	155
Table 12-70 Secondary vtable for basic_ostream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	156
Table 12-71 VTT for basic_ostream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	156
Table 12-72 libstdc++ - Class basic_ostream<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces	157
Table 12-73 Primary vtable for basic_stringbuf<char, char_traits<char>, allocator<char> >	157
Table 12-74 typeid for basic_stringbuf<char, char_traits<char>, allocator<char> >	158
Table 12-75 libstdc++ - Class basic_stringbuf<char, char_traits<char>, allocator<char> > Function Interfaces.....	158
Table 12-76 Primary vtable for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	159
Table 12-77 typeid for basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> >	160
Table 12-78 libstdc++ - Class basic_stringbuf<wchar_t, char_traits<wchar_t>, allocator<wchar_t> > Function Interfaces	161
Table 12-79 Primary vtable for basic_istream<char, char_traits<char> >	161
Table 12-80 Secondary vtable for basic_istream<char, char_traits<char> >	161
Table 12-81 Secondary vtable for basic_istream<char, char_traits<char> >	162
Table 12-82 VTT for basic_istream<char, char_traits<char> >	162
Table 12-83 libstdc++ - Class basic_istream<char, char_traits<char> > Function Interfaces	162
Table 12-84 Primary vtable for basic_istream<wchar_t, char_traits<wchar_t> >	163
Table 12-85 Secondary vtable for basic_istream<wchar_t, char_traits<wchar_t> >	163
Table 12-86 Secondary vtable for basic_istream<wchar_t, char_traits<wchar_t> >	163
Table 12-87 VTT for basic_istream<wchar_t, char_traits<wchar_t> >	164
Table 12-88 libstdc++ - Class basic_istream<wchar_t, char_traits<wchar_t> >	

Function Interfaces.....	164
Table 12-89 Primary vtable for basic_istream<char, char_traits<char> >.....	164
Table 12-90 Secondary vtable for basic_istream<char, char_traits<char> >.....	164
Table 12-91 VTT for basic_istream<char, char_traits<char> >.....	165
Table 12-92 libstdcxx - Class basic_istream<char, char_traits<char> > Function Interfaces.....	165
Table 12-93 Primary vtable for basic_istream<wchar_t, char_traits<wchar_t> >.....	166
Table 12-94 Secondary vtable for basic_istream<wchar_t, char_traits<wchar_t> >.....	166
Table 12-95 VTT for basic_istream<wchar_t, char_traits<wchar_t> >.....	167
Table 12-96 libstdcxx - Class basic_istream<wchar_t, char_traits<wchar_t> > Function Interfaces.....	167
Table 12-97 Primary vtable for basic_ostream<char, char_traits<char> >.....	168
Table 12-98 Secondary vtable for basic_ostream<char, char_traits<char> >.....	168
Table 12-99 VTT for basic_ostream<char, char_traits<char> >.....	169
Table 12-100 libstdcxx - Class basic_ostream<char, char_traits<char> > Function Interfaces.....	169
Table 12-101 Primary vtable for basic_ostream<wchar_t, char_traits<wchar_t> >.....	169
Table 12-102 Secondary vtable for basic_ostream<wchar_t, char_traits<wchar_t> >.....	170
Table 12-103 VTT for basic_ostream<wchar_t, char_traits<wchar_t> >.....	170
Table 12-104 libstdcxx - Class basic_ostream<wchar_t, char_traits<wchar_t> > Function Interfaces.....	170
Table 12-105 Primary vtable for basic_fstream<char, char_traits<char> >.....	171
Table 12-106 Secondary vtable for basic_fstream<char, char_traits<char> >.....	171
Table 12-107 Secondary vtable for basic_fstream<char, char_traits<char> >.....	171
Table 12-108 VTT for basic_fstream<char, char_traits<char> >.....	172
Table 12-109 libstdcxx - Class basic_fstream<char, char_traits<char> > Function Interfaces.....	172
Table 12-110 Primary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >.....	172
Table 12-111 Secondary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >.....	173
Table 12-112 Secondary vtable for basic_fstream<wchar_t, char_traits<wchar_t> >.....	173
Table 12-113 VTT for basic_fstream<wchar_t, char_traits<wchar_t> >.....	173
Table 12-114 libstdcxx - Class basic_fstream<wchar_t, char_traits<wchar_t> > Function Interfaces.....	174
Table 12-115 Primary vtable for basic_ifstream<char, char_traits<char> >.....	174
Table 12-116 Secondary vtable for basic_ifstream<char, char_traits<char> >.....	174
Table 12-117 VTT for basic_ifstream<char, char_traits<char> >.....	175
Table 12-118 libstdcxx - Class basic_ifstream<char, char_traits<char> > Function Interfaces.....	175
Table 12-119 Primary vtable for basic_ifstream<wchar_t, char_traits<wchar_t> >.....	175
Table 12-120 Secondary vtable for basic_ifstream<wchar_t, char_traits<wchar_t> >.....	175
Table 12-121 VTT for basic_ifstream<wchar_t, char_traits<wchar_t> >.....	176
Table 12-122 libstdcxx - Class basic_ifstream<wchar_t, char_traits<wchar_t> > Function Interfaces.....	176
Table 12-123 Primary vtable for basic_ofstream<char, char_traits<char> >.....	176
Table 12-124 Secondary vtable for basic_ofstream<char, char_traits<char> >.....	177
Table 12-125 VTT for basic_ofstream<char, char_traits<char> >.....	177

Table 12-126 libstdcxx - Class basic_ofstream<char, char_traits<char> > Function Interfaces	177
Table 12-127 Primary vtable for basic_ofstream<wchar_t, char_traits<wchar_t> >	178
Table 12-128 Secondary vtable for basic_ofstream<wchar_t, char_traits<wchar_t> >	178
Table 12-129 VTT for basic_ofstream<wchar_t, char_traits<wchar_t> >	178
Table 12-130 libstdcxx - Class basic_ofstream<wchar_t, char_traits<wchar_t> > Function Interfaces	179
Table 12-131 Primary vtable for basic_streambuf<char, char_traits<char> >	179
Table 12-132 typeinfo for basic_streambuf<char, char_traits<char> >	180
Table 12-133 libstdcxx - Class basic_streambuf<char, char_traits<char> > Function Interfaces	180
Table 12-134 Primary vtable for basic_streambuf<wchar_t, char_traits<wchar_t> >	181
Table 12-135 typeinfo for basic_streambuf<wchar_t, char_traits<wchar_t> >	182
Table 12-136 libstdcxx - Class basic_streambuf<wchar_t, char_traits<wchar_t> > Function Interfaces	183
Table 12-137 Primary vtable for basic_filebuf<char, char_traits<char> >	183
Table 12-138 typeinfo for basic_filebuf<char, char_traits<char> >	184
Table 12-139 libstdcxx - Class basic_filebuf<char, char_traits<char> > Function Interfaces	184
Table 12-140 Primary vtable for basic_filebuf<wchar_t, char_traits<wchar_t> >	185
Table 12-141 typeinfo for basic_filebuf<wchar_t, char_traits<wchar_t> >	186
Table 12-142 libstdcxx - Class basic_filebuf<wchar_t, char_traits<wchar_t> > Function Interfaces	186
Table 12-143 typeinfo for ios_base	187
Table 12-144 typeinfo for basic_ios<wchar_t, char_traits<wchar_t> >	188
Table 12-145 typeinfo for ios_base::failure	188
Table 12-146 typeinfo for __timepunct<char>	189
Table 12-147 libstdcxx - Class __timepunct<char> Function Interfaces	189
Table 12-148 typeinfo for __timepunct<wchar_t>	190
Table 12-149 libstdcxx - Class __timepunct<wchar_t> Function Interfaces	190
Table 12-150 typeinfo for messages_base	190
Table 12-151 libstdcxx - Class messages<char> Function Interfaces	191
Table 12-152 libstdcxx - Class messages<wchar_t> Function Interfaces	191
Table 12-153 typeinfo for messages_byname<char>	192
Table 12-154 libstdcxx - Class messages_byname<char> Function Interfaces	192
Table 12-155 typeinfo for messages_byname<wchar_t>	192
Table 12-156 libstdcxx - Class messages_byname<wchar_t> Function Interfaces	192
Table 12-157 typeinfo for numpunct<char>	193
Table 12-158 libstdcxx - Class numpunct<char> Function Interfaces	193
Table 12-159 typeinfo for numpunct<wchar_t>	193
Table 12-160 libstdcxx - Class numpunct<wchar_t> Function Interfaces	194
Table 12-161 typeinfo for numpunct_byname<char>	194
Table 12-162 libstdcxx - Class numpunct_byname<char> Function Interfaces	194
Table 12-163 typeinfo for numpunct_byname<wchar_t>	195
Table 12-164 libstdcxx - Class numpunct_byname<wchar_t> Function Interfaces	195
Table 12-165 typeinfo for codecvt_base	196
Table 12-166 Primary vtable for codecvt<char, char, __mbstate_t>	196
Table 12-167 typeinfo for codecvt<char, char, __mbstate_t>	197

Table 12-168 libstdcxx - Class codecvt<char, char, __mbstate_t> Function Interfaces	197
Table 12-169 Primary vtable for codecvt<wchar_t, char, __mbstate_t>	198
Table 12-170 typeinfo for codecvt<wchar_t, char, __mbstate_t>	199
Table 12-171 libstdcxx - Class codecvt<wchar_t, char, __mbstate_t> Function Interfaces	199
Table 12-172 Primary vtable for codecvt_byname<char, char, __mbstate_t>	199
Table 12-173 typeinfo for codecvt_byname<char, char, __mbstate_t>	200
Table 12-174 libstdcxx - Class codecvt_byname<char, char, __mbstate_t> Function Interfaces	201
Table 12-175 Primary vtable for codecvt_byname<wchar_t, char, __mbstate_t>	201
Table 12-176 typeinfo for codecvt_byname<wchar_t, char, __mbstate_t>	202
Table 12-177 libstdcxx - Class codecvt_byname<wchar_t, char, __mbstate_t> Function Interfaces	202
Table 12-178 typeinfo for collate<char>	202
Table 12-179 libstdcxx - Class collate<char> Function Interfaces	203
Table 12-180 typeinfo for collate<wchar_t>	203
Table 12-181 libstdcxx - Class collate<wchar_t> Function Interfaces	203
Table 12-182 typeinfo for collate_byname<char>	204
Table 12-183 libstdcxx - Class collate_byname<char> Function Interfaces	204
Table 12-184 typeinfo for collate_byname<wchar_t>	204
Table 12-185 libstdcxx - Class collate_byname<wchar_t> Function Interfaces	205
Table 12-186 typeinfo for time_base	205
Table 12-187 typeinfo for time_get_byname<char, istreambuf_iterator<char, char_traits<char> > >	205
Table 12-188 libstdcxx - Class time_get_byname<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces	206
Table 12-189 typeinfo for time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >	206
Table 12-190 libstdcxx - Class time_get_byname<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces	207
Table 12-191 typeinfo for time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > >	207
Table 12-192 libstdcxx - Class time_put_byname<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces	207
Table 12-193 typeinfo for time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >	208
Table 12-194 libstdcxx - Class time_put_byname<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces	208
Table 12-195 libstdcxx - Class time_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces	209
Table 12-196 libstdcxx - Class time_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces	209
Table 12-197 typeinfo for time_put<char, ostreambuf_iterator<char, char_traits<char> > >	210
Table 12-198 libstdcxx - Class time_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces	210
Table 12-199 typeinfo for time_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >	211
Table 12-200 libstdcxx - Class time_put<wchar_t, ostreambuf_iterator<wchar_t,	

char_traits<wchar_t> > > Function Interfaces	211
Table 12-201 libstdcxx - Class moneypunct<char, false> Function Interfaces	212
Table 12-202 libstdcxx - Class moneypunct<char, true> Function Interfaces	212
Table 12-203 libstdcxx - Class moneypunct<wchar_t, false> Function Interfaces	213
Table 12-204 libstdcxx - Class moneypunct<wchar_t, true> Function Interfaces	214
Table 12-205 typeinfo for moneypunct_byname<char, false>	214
Table 12-206 libstdcxx - Class moneypunct_byname<char, false> Function Interfaces	214
Table 12-207 typeinfo for moneypunct_byname<char, true>	215
Table 12-208 libstdcxx - Class moneypunct_byname<char, true> Function Interfaces	215
Table 12-209 typeinfo for moneypunct_byname<wchar_t, false>	215
Table 12-210 libstdcxx - Class moneypunct_byname<wchar_t, false> Function Interfaces	216
Table 12-211 typeinfo for moneypunct_byname<wchar_t, true>	216
Table 12-212 libstdcxx - Class moneypunct_byname<wchar_t, true> Function Interfaces	216
Table 12-213 typeinfo for money_base	217
Table 12-214 typeinfo for money_get<char, istreambuf_iterator<char, char_traits<char> > >	217
Table 12-215 libstdcxx - Class money_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces	217
Table 12-216 typeinfo for money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >	218
Table 12-217 libstdcxx - Class money_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces	218
Table 12-218 typeinfo for money_put<char, ostreambuf_iterator<char, char_traits<char> > >	219
Table 12-219 libstdcxx - Class money_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces	219
Table 12-220 typeinfo for money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >	219
Table 12-221 libstdcxx - Class money_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces	220
Table 12-222 libstdcxx - Class locale Function Interfaces	220
Table 12-223 typeinfo for locale::facet	220
Table 12-224 typeinfo for num_get<char, istreambuf_iterator<char, char_traits<char> > >	221
Table 12-225 libstdcxx - Class num_get<char, istreambuf_iterator<char, char_traits<char> > > Function Interfaces	222
Table 12-226 typeinfo for num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > >	222
Table 12-227 libstdcxx - Class num_get<wchar_t, istreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces	222
Table 12-228 typeinfo for num_put<char, ostreambuf_iterator<char, char_traits<char> > >	223
Table 12-229 libstdcxx - Class num_put<char, ostreambuf_iterator<char, char_traits<char> > > Function Interfaces	223
Table 12-230 typeinfo for num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > >	224

Table 12-231 libstdcxx - Class num_put<wchar_t, ostreambuf_iterator<wchar_t, char_traits<wchar_t> > > Function Interfaces.....	224
Table 12-232 libstdcxx - Class gsllice Function Interfaces.....	225
Table 12-233 libstdcxx - Class __basic_file<char> Function Interfaces.....	225
Table 12-234 libstdcxx - Class valarray<unsigned int> Function Interfaces.....	226
Table 12-235 libstdcxx - Class __gnu_cxx::__pool<true> Function Interfaces	226
Table 12-236 libstdcxx - Class __gnu_cxx::__pool<false> Function Interfaces.....	227
Table 12-237 libstdcxx - Class __gnu_cxx::free_list Function Interfaces	227
Table 12-238 libstdcxx - Class locale::_Impl Function Interfaces	227
Table 12-239 libstdcxx - Namespace std Functions Function Interfaces	227
Table A-1 libc Function Interfaces	231
Table A-2 libc Data Interfaces.....	247
Table A-3 libcrypt Function Interfaces	247
Table A-4 libdl Function Interfaces	247
Table A-5 libgcc_s Function Interfaces.....	247
Table A-6 libm Function Interfaces	248
Table A-7 libm Data Interfaces.....	253
Table A-8 libpthread Function Interfaces.....	253
Table A-9 librt Function Interfaces	256
Table A-10 libutil Function Interfaces	257

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

The committee responsible for this document is Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 22, Programming languages, their environments and system software interfaces.

This document is a direct adoption of the Linux Standard Base (LSB) 5.0 Common Definitions, issued by the Linux Foundation. The previous release of these standards, ISO/IEC 23360-1 through ISO/IEC 23360-8:2006 were international standards published under the ISO/IEC/JTC 1 Publicly Available Specification process. This document, and others in the series, are published under the GNU Free Documentation License (See Annex B).

This is version 1.0 of the Linux Standard Base (LSB) core specifications for AMD64 (X86-64) architecture. This standard replaces the core specification portion of ISO/IEC 23360-4:2006 Linux Standard Base, which is cancelled and replaced by ISO/IEC 23360-4-2 through ISO/IEC 23360-4-3. The general parts and the processor specific parts of the original Linux Standard Base are also subdivided as follows:

- The common definitions ISO/IEC 23360-1-1;
- The core specification generic part ISO/IEC 23360-1-2;
- The desktop specification generic part ISO/IEC 23360-1-3;
- The languages specification generic part ISO/IEC 23360-1-4;
- The imaging specification generic part ISO/IEC 23360-1-5;
- The Intel X86-32 architecture core and desktop specifications in ISO/IEC 23360-2-2 and ISO/IEC 23360-2-3 respectively;
- The Intel IA64 (Itanium) architecture core and desktop specification in ISO/IEC 23360-3-2 and ISO/IEC 23360-3-3 respectively;
- The AMD64 (X86-64) architecture core and desktop specification in ISO/IEC 23360-4-2 (this document) and ISO/IEC 23360-4-3 respectively;

- The PowerPC 32 architecture core and desktop specification in ISO/IEC 23360-5-2 and ISO/IEC 23360-5-3 respectively;
- The PowerPC 64 architecture core and desktop specification in ISO/IEC 23360-6-2 and ISO/IEC 23360-6-3 respectively;
- The IBM S390 architecture core and desktop specification in ISO/IEC 23360-7-2 and ISO/IEC 23360-7-3 respectively; and
- The IBM S390X architecture core and desktop specification in ISO/IEC 23360-8-2 and ISO/IEC 23360-8-3 respectively.

Throughout this document, there are many clauses where the generic specification of the component being specified is in ISO/IEC 23360-1-2. To find the corresponding generic text, a search of ISO/IEC 23360-1-2 with the clause title as text will take you to the corresponding generic clause.

Status of this Document

This is a released specification, version 5.0. Other documents may supersede or augment this specification.

A list of current released Linux Standard Base (LSB) specifications is available at <http://refspecs.linuxbase.org> (<http://refspecs.linuxbase.org/>).

If you wish to make comments regarding this document in a manner that is tracked by the LSB project, please submit them using our public bug database at <http://bugs.linuxbase.org>. Please enter your feedback, carefully indicating the title of the section for which you are submitting feedback, and the volume and version of the specification where you found the problem, quoting the incorrect text if appropriate. If you are suggesting a new feature, please indicate what the problem you are trying to solve is. That is more important than the solution, in fact.

If you do not have or wish to create a bug database account then you can also e-mail feedback to [<lsb-discuss@lists.linuxfoundation.org>](mailto:lsb-discuss@lists.linuxfoundation.org) (subscribe (<http://lists.linuxfoundation.org/mailman/listinfo/lsb-discuss>), archives (<http://lists.linuxfoundation.org/pipermail/lsb-discuss/>)), and arrangements will be made to transpose the comments to our public bug database.

Introduction

The LSB defines a binary interface for application programs that are compiled and packaged for LSB-conforming implementations on many different hardware architectures. A binary specification must include information specific to the computer processor architecture for which it is intended. To avoid the complexity of conditional descriptions, the specification has instead been divided into generic parts which are augmented by one of several architecture-specific parts, depending on the target processor architecture; the generic part will indicate when reference must be made to the architecture part, and vice versa.

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

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

1. The first number (x) is the major version number. Versions sharing the same major version number shall be compatible in a backwards direction; that is, a newer version shall be compatible with an older version. Any deletion of a library results in a new major version number. Interfaces marked as deprecated may be removed from the specification at a major version change.
2. The second number (y) is the minor version number. Libraries and individual interfaces may be added, but not removed. Interfaces may be marked as deprecated at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.
3. The third number (z), if present, is the editorial level. Only editorial changes should be included in such versions.

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

LSB is a trademark of the Linux Foundation. Developers of applications or implementations interested in using the trademark should see the Linux Foundation Certification Policy for details.

I Introductory Elements

1 Scope

1.1 General

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

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

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

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

1.2 Module Specific Scope

This is the X86-64 architecture specific part of the Core module of the Linux Standard Base (LSB). This part supplements the common part of the LSB Core module with those interfaces that differ between architectures.

This part should be used in conjunction with LSB Core - Generic, the common part. Whenever a section of the common part is supplemented by architecture-specific information, the common part includes a reference to the architecture-specific part. This part may also contain additional information that is not referenced in the common part.

Interfaces described in this part of the LSB Core Specification are mandatory except where explicitly listed otherwise. Interfaces described in the LSB Core module are supplemented by other LSB modules. All other modules depend on the presence of LSB Core.

2 References

2.1 Normative References

The following specifications are incorporated by reference into this specification. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced specification (including any amendments) applies.

Note: Where copies of a referenced specification are available on the World Wide Web, a Uniform Resource Locator (URL) is given, for informative purposes only. Such URL might at any given time resolve to a more recent copy of the specification, or be out of date (not resolve). Reference copies of specifications at the revision level indicated may be found at the Linux Foundation's Reference Specifications (<http://refspecs.linuxbase.org>) site.

Table 2-1 Normative References

Name	Title	URL
LSB Core - Generic	Linux Standard Base - Core Specification - Generic	http://www.linuxbase.org/spec/
AMD64 Architecture Programmer's Manual, Volume 1	AMD64 Architecture Programmer's Manual, Volume 1: Application Programming 24592 3.08	http://www.amd.com/us-en/Processors/DevelopWithAMD/
AMD64 Architecture Programmer's Manual, Volume 2	AMD64 Architecture Programmer's Manual, Volume 2: System Programming 24593 3.08	http://www.amd.com/us-en/Processors/DevelopWithAMD/
AMD64 Architecture Programmer's Manual, Volume 3	AMD64 Architecture Programmer's Manual, Volume 3: General Purpose and System Instructions 24594 3.03	http://www.amd.com/us-en/Processors/DevelopWithAMD/
AMD64 Architecture Programmer's Manual, Volume 4	AMD64 Architecture Programmer's Manual, Volume 4: 128-bit Media Instructions 26568 3.04	http://www.amd.com/us-en/Processors/DevelopWithAMD/
AMD64 Architecture Programmer's Manual, Volume 5	AMD64 Architecture Programmer's Manual, Volume 5: 64-bit Media and x87 Floating-Point Instructions 26569 3.03	http://www.amd.com/us-en/Processors/DevelopWithAMD/
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 3.0	http://refspecs.linuxbase.org/fhs

Name	Title	URL
ISO C (1999)	ISO/IEC 9899:1999 - Programming Languages -- C	
ISO/IEC 14882: 2003 C++ Language	ISO/IEC 14882: 2003 Programming languages --C++	
Itanium™ C++ ABI	Itanium™ C++ ABI (Revision 1.86)	http://refspecs.linuxfoundation.org/cxxabi-1.86.html
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
Libcursesw API	Libcursesw API	http://invisible-island.net/ncurses/man/ncurses.3x.html
Libcursesw Placeholder	Libcursesw Specification Placeholder	http://refspecs.linux-foundation.org/libcursesw/libcurses.html
POSIX 1003.1-2001 (ISO/IEC 9945-2003)	ISO/IEC 9945-1:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 1: Base Definitions ISO/IEC 9945-2:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces ISO/IEC 9945-3:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 3: Shell and Utilities ISO/IEC 9945-4:2003 Information technology -- Portable Operating System Interface (POSIX) -- Part 4: Rationale Including Technical Cor. 1: 2004	http://www.unix.org/version3/

Name	Title	URL
POSIX 1003.1-2008 (ISO/IEC 9945-2009)	Portable Operating System Interface (POSIX®) 2008 Edition / The Open Group Technical Standard Base Specifications, Issue 7	http://www.unix.org/version4/
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
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	http://refspecs.linuxfoundation.org/svid4/
System V ABI	System V Application Binary Interface, Edition 4.1	http://www.sco.com/developers/devspecs/gabi41.pdf
System V ABI Update	System V Application Binary Interface - DRAFT - 17 December 2003	http://www.sco.com/developers/gabi/2003-12-17/contents.html
System V Application Binary Interface AMD64 Architecture Processor Supplement	System V Application Binary Interface AMD64 Architecture Processor Supplement, Draft Version 0.95	http://refspecs.linux-foundation.org/elf/x86_64-abi-0.95.pdf
X/Open Curses, Issue 7	X/Open Curses, Issue 7 (ISBN: 1-931624-83-6, The Open Group, November 2009)	https://www2.opengroup.org/ogsys/catalog/C094

2.2 Informative References/Bibliography

The documents listed below provide essential background information to implementors of this specification. These references are included for information only, and do not represent normative parts of this specification.

Table 2-2 Other References

Name	Title	URL
DWARF Debugging Information Format, Version 4	DWARF Debugging Information Format, Version 4 (June 10, 2010)	http://www.dwarfstd.org/doc/DWARF4.pdf
IEC 60559/IEEE 754 Floating Point	IEC 60559:1989 Binary floating-point arithmetic for microprocessor systems	http://www.ieee.org/
ISO/IEC TR14652	ISO/IEC Technical Report 14652:2002 Specification method for cultural conventions	
ITU-T V.42	International Telecommunication Union Recommendation V.42 (2002): Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion ITUV	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Li18nux Globalization Specification	LI18N 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.openi18n.org/docs/html/LI18N-2000-amd4.htm
Linux Allocated Device Registry	LINUX ALLOCATED DEVICES	http://www.lanana.org/docs/device-list/devices-2.6+.txt
Linux Assigned Names And Numbers Authority	Linux Assigned Names And Numbers Authority	http://www.lanana.org/
Mozilla's NSS SSL Reference	Mozilla's NSS SSL Reference	http://www.mozilla.org/projects/security/pki/nss/ref/ssl/
NSPR Reference	Mozilla's NSPR Reference	http://refspecs.linuxfoundation.org/NSPR_API_Reference/NSPR_API.html
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

Name	Title	URL
RFC 1321: The MD5 Message-Digest Algorithm	IETF RFC 1321: The MD5 Message-Digest Algorithm	http://www.ietf.org/rfc/rfc1321.txt
RFC 1833: Binding Protocols for ONC RPC Version 2	IETF RFC 1833: Binding Protocols for ONC RPC Version 2	http://www.ietf.org/rfc/rfc1833.txt
RFC 1950: ZLIB Compressed Data Format Specification	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821: Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822: Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 5531/4506 RPC & XDR	IETF RFC 5531 & 4506	http://www.ietf.org/
RFC 791: Internet Protocol	IETF RFC 791: Internet Protocol Specification	http://www.ietf.org/rfc/rfc791.txt
RPM Package Format	RPM Package Format V3.0	http://www.rpm.org/max-rpm/s1-rpm-file-format-rpm-file-format.html
zlib Manual	zlib 1.2 Manual	http://www.gzip.org/zlib/

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on x86-64 Linux Standard Base systems, with the specified runtime names. These names override or supplement the names specified in the generic LSB (LSB Core - Generic) specification. The specified program interpreter, referred to as `proginterp` in this table, shall be used to load the shared libraries specified by `DT_NEEDED` entries at run time.

Table 3-1 Standard Library Names

Library	Runtime Name
libc	libc.so.6
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libgcc_s	libgcc_s.so.1
libm	libm.so.6
libncurses	libncurses.so.5
libncursesw	libncursesw.so.5
libpthread	libpthread.so.0
libstdc++	libstdc++.so.6
libutil	libutil.so.1
libz	libz.so.1
proginterp	/lib64/ld-lsb-x86-64.so.3

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

3.2 LSB Implementation Conformance

A conforming implementation is necessarily architecture specific, and must provide the interfaces specified by both the generic LSB Core specification (LSB Core - Generic) and the relevant architecture specific part of the LSB Core Specification.

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

A conforming implementation shall satisfy the following requirements:

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

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

3.3 LSB Application Conformance

A conforming application containing object files is necessarily architecture specific, and must conform to both the generic LSB Core specification (LSB Core - Generic) and the relevant architecture specific part of the LSB Core Specification. A conforming application which contains no object files may be architecture neutral. Architecture neutral applications shall conform only to the requirements of the generic LSB Core specification (LSB Core - Generic).

A conforming application shall satisfy the following requirements:

- Executable files shall be either object files in the format defined in the Object Format section of this specification, or script files in a scripting language where the interpreter is required by this specification.
- Object files shall participate in dynamic linking as defined in the Program Loading and Linking section of this specification.

- Object files shall employ only the instructions, traps, and other low-level facilities defined as being for use by applications in the Low-Level System Information section of this specification
- If the application requires any optional interface defined in this specification in order to be installed or to execute successfully, the requirement for that optional interface shall be stated in the application's documentation.
- The application shall not use any interface or data format that is not required to be provided by a conforming implementation, unless such an interface or data format is supplied by another application through direct invocation of that application during execution. The other application must also be a conforming application, and the use of such interface or data format, as well as its source (in other words, the other conforming application), shall be identified in the documentation of the application.
- The application shall not use any values for a named interface that are reserved for vendor extensions.

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

4 Terms and Definitions

For the purposes of this document, the terms given in *ISO/IEC Directives, Part 2, Annex H* and the following apply.

archLSB

Some LSB specification documents have both a generic, architecture-neutral part and an architecture-specific part. The latter describes elements whose definitions may be unique to a particular processor architecture. The term archLSB may be used in the generic part to refer to the corresponding section of the architecture-specific part.

Binary Standard, ABI

The total set of interfaces that are available to be used in the compiled binary code of a conforming application, including the run-time details such as calling conventions, binary format, C++ name mangling, etc.

Implementation-defined

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

Shell Script

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

Source Standard, API

The total set of interfaces that are available to be used in the source code of a conforming application. Due to translations, the Binary Standard and the Source Standard may contain some different interfaces.

Undefined

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

Unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or

behavior cannot be assured to be portable across conforming implementations.

In addition, for the portions of this specification which build on IEEE Std 1003.1-2001, the definitions given in *IEEE Std 1003.1-2001, Base Definitions, Chapter 3* apply.

5 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) [SUSv4]

refers to the interface named `forkpty()` with symbol version `GLIBC_2.0` that is defined in the reference indicated by the tag `SUSv4`.

Note: For symbols with versions which differ between architectures, the symbol versions are defined in the architecture specific parts of of this module specification only. In the generic part, they will appear without symbol versions.

II Executable and Linking Format (ELF)

6 Introduction

Executable and Linking Format (ELF) defines the object format for compiled applications. This specification supplements the information found in System V ABI Update and System V Application Binary Interface AMD64 Architecture Processor Supplement, and is intended to document additions made since the publication of that document.

7 Low Level System Information

7.1 Machine Interface

7.1.1 Processor Architecture

The AMD64 Architecture is specified by the following documents

- AMD64 Architecture Programmer's Manual, Volume 1
- AMD64 Architecture Programmer's Manual, Volume 2
- AMD64 Architecture Programmer's Manual, Volume 3
- AMD64 Architecture Programmer's Manual, Volume 4
- AMD64 Architecture Programmer's Manual, Volume 5
- System V Application Binary Interface AMD64 Architecture Processor Supplement

Applications conforming to this specification must provide feedback to the user if a feature that is required for correct execution of the application is not present. Applications conforming to this specification should attempt to execute in a diminished capacity if a required instruction set feature is not present. In particular, applications should not rely on the availability of the 3DNow!™ technology. In addition, a conforming application shall not use any instruction from Table 7-1.

Note: While this specification carries the attribution "AMD64", it is intended to apply to the entire $\times 86_64$ set of processors, including those based on Intel® 64 Architecture. However, this specification defers to the AMD64 architecture specifications listed above.

Table 7-1 Non Conforming Instructions

LAHF	SAHF
SYSCALL	SYSRET
SYSENTER	SYSEXIT
CMPXCHG16B	FFXSR

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

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

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

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

7.1.2 Data Representation

7.1.2.1 Introduction

LSB-conforming applications shall use the data representation as defined in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

Note: The System V Application Binary Interface AMD64 Architecture Processor Supplement specification is itself layered on top of the System V Application Binary Interface - Intel386™ Architecture Processor Supplement.

7.1.2.2 Byte Ordering

LSB-conforming applications shall use the byte ordering defined in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.1.2.3 Fundamental Types

LSB-conforming applications shall use only the fundamental types described in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.1.2.4 Aggregates and Unions

LSB-conforming applications shall use alignment for aggregates and unions as described in Section 3.1.2 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.1.2.5 Bit Fields

LSB-conforming applications utilizing bit-fields shall follow the requirements of Section 3.1.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.2 Function Calling Sequence

7.2.1 Introduction

LSB-conforming applications shall use only the following features of the function calling sequence as defined in Section 3.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.2.2 Registers

LSB-conforming applications shall use only the registers described in Section 3.2.1 (Registers and the Stack Frame) of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.2.3 Floating Point Registers

LSB-conforming applications shall use only the floating point registers described in Section 3.2.1 (Registers and the Stack Frame) of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.2.4 Stack Frame

LSB-conforming applications shall use stack frames as described in Section 3.2.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.2.5 Arguments

LSB-conforming applications shall pass parameters to functions as described in Section 3.2.3 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.2.6 Return Values

Values are returned from functions as described in Section 3.3.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.3 Operating System Interface

LSB-conforming applications shall use only the following features of the Operating System Interfaces as defined in Section 3.3 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.3.1 Exception Interface

Synchronous and floating point or coprocessor exceptions shall behave as described in Section 3.3.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.3.2 Virtual Address Space

LSB-Conforming applications shall use only the virtual address space described in Section 3.3.2 and 3.3.4 of the System V Application Binary Interface AMD64 Architecture Processor Supplement. Virtual memory page sizes shall be subject to the limitations described in Section 3.3.3 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.4 Process Initialization

LSB-conforming applications shall use only the following features of the Process Initialization as defined in Section 3.4 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.4.1 Special Registers

During process initialization, the special registers shall be initialized as described in Section 3.4.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.4.2 Process Stack (on entry)

The process stack shall be initialized as described in Section 3.4.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.4.3 Auxiliary Vector

The auxiliary vector shall be initialized as described in Section 3.4.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.5 Coding Examples

LSB-conforming applications may use the coding examples given in Section 3.5 of the System V Application Binary Interface AMD64 Architecture Processor Supplement to guide implementation of fundamental operations in the following areas.

7.5.1 Code Model Overview/Architecture Constraints

Section 3.5.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement describes a number of code models. LSB-Conforming applications may use any of these models except the Kernel and Large code models.

7.5.2 Position-Independent Function Prologue

LSB-conforming applications may follow the position-independent function prologue example in Section 3.5.3 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.5.3 Data Objects

LSB-conforming applications may follow the data objects examples in Section 3.5.4 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.5.4 Function Calls

LSB-conforming applications may follow the function call examples in Section 3.5.5 of the System V Application Binary Interface AMD64 Architecture Processor Supplement. See Chapter 3 of System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.5.5 Branching

LSB-conforming applications may follow the branching examples in Section 3.5.6 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.6 C Stack Frame

7.6.1 Variable Argument List

LSB-Conforming applications shall only use variable arguments to functions in the manner described in Section 3.5.7 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

7.7 Debug Information

LSB-Conforming applications may include DWARF debugging information. The DWARF Release Number and Register Number Mapping shall be as described in Section 3.6 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8 Object Format

8.1 Introduction

LSB-conforming implementations shall support the Executable and Linking Format (ELF) object file format, as defined by the following documents:

- System V ABI
- System V ABI Update
- System V Application Binary Interface AMD64 Architecture Processor Supplement
- LSB Core - Generic
- this document

8.2 ELF Header

8.2.1 Machine Information

LSB-conforming applications shall identify the Machine Information as defined in Section 4.1.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.3 Sections

8.3.1 Introduction

In addition to the requirements for ELF sections described in the generic LSB Core specification, conforming implementations shall support architecture specific sections as described below.

Note: The System V Application Binary Interface AMD64 Architecture Processor Supplement specifies some architecture specific section flags and section types that are not required by LSB-conforming systems.

8.3.2 Special Sections

The following architecture-specific sections are defined in the System V Application Binary Interface AMD64 Architecture Processor Supplement.

Table 8-1 ELF Special Sections

Name	Type	Attributes
.got	SHT_PROGBITS	SHF_AL- LOC+SHF_WRITE
.plt	SHT_PROGBITS	SHF_ALLOC+SHF_EX- ECINSTR

.got

This section holds the global offset table.

.plt

This section holds the procedure linkage table.

Note: Since LSB-conforming implementations are not required to support the large code model, it is not necessary for them to provide support for the additional special sections for the large code model described in the System V Application Binary Interface AMD64 Architecture Processor Supplement.

Also, the System V Application Binary Interface AMD64 Architecture Processor Supplement specifies a section `.eh_frame`, with a type of `SHT_AMD64_UNWIND`. This section is described in the generic LSB-Core specification, but with type `SHT_PROGBITS`. This specification does not require support for the `SHT_AMD64_UNWIND` section type.

8.3.3 Additional Special Sections

The following additional sections are defined here.

Table 8-2 Additional Special Sections

Name	Type	Attributes
<code>.rela.dyn</code>	<code>SHT_RELA</code>	<code>SHF_ALLOC</code>
<code>.rela.plt</code>	<code>SHT_RELA</code>	<code>SHF_ALLOC</code>

`.rela.dyn`

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

`.rela.plt`

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

8.4 Symbol Table

LSB-conforming applications shall use Symbol Tables as defined in Section 4.3 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

8.5 Relocation

LSB-conforming implementation shall support the required relocation types defined in Section 4.4.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

Note: Since LSB-conforming implementations are not required to support the large code model, it is not necessary for them to provide support for the additional relocation types for the large code model described in the System V Application Binary Interface AMD64 Architecture Processor Supplement.

9 Program Loading and Dynamic Linking

9.1 Introduction

LSB-conforming implementations shall support the object file information and system actions that create running programs as specified in the System V ABI , System V ABI Update , System V Application Binary Interface AMD64 Architecture Processor Supplement and as supplemented by the generic LSB specification and LSB Core - Generic.

9.2 Program Header

LSB-conforming implementations are not required to support the additional types and flags for this architecture as defined in Section 5.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

Note: The System V Application Binary Interface AMD64 Architecture Processor Supplement specification is itself layered on top of the System V Application Binary Interface - Intel386™ Architecture Processor Supplement. As such, the requirements of that specification are still requirements of this specification.

9.3 Program Loading

LSB-conforming implementations shall map file pages to virtual memory pages as described in Section 5.1 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

9.4 Dynamic Linking

9.4.1 Introduction

LSB-conforming implementations shall provide dynamic linking as specified in Section 5.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement, except as described in the following sections.

Note: Since LSB-conforming implementations are not required to support the large model, support for dynamic linking of large model code is not required.

9.4.2 Dynamic Section

Dynamic section entries give information to the dynamic linker. The following dynamic entry types shall be supported:

DT_JMPREL

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

DT_PLTGOT

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

DT_RELACOUNT

The number of relative relocations in .rela.dyn

9.4.3 Global Offset Table

LSB-conforming implementations shall support a Global Offset Table as described in Section 5.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

9.4.4 Function Addresses

Function addresses shall behave as described in Section 5.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

9.4.5 Procedure Linkage Table

LSB-conforming implementations shall support a Procedure Linkage Table as described in Section 5.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

9.4.6 Initialization and Termination Functions

LSB-conforming implementations shall support initialization and termination functions as specified in Section 5.2.2 of the System V Application Binary Interface AMD64 Architecture Processor Supplement.

III Base Libraries

10 Libraries

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

Only interfaces and interface details which are unique to the X86-64 platform are defined here. This section should be used in conjunction with the corresponding section of LSB Core - Generic.

10.1 Program Interpreter/Dynamic Linker

The Program Interpreter shall be `/lib64/ld-lsb-x86-64.so.3`.

10.2 Interfaces for libc

Table 10-1 defines the library name and shared object name for the libc library

Table 10-1 libc Definition

Library:	libc
SONAME:	libc.so.6

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

[LFS] Large File Support
 [LSB] LSB Core - Generic
 [RPC + XDR] RFC 5531/4506 RPC & XDR
 [SUSv2] SUSv2
 [SUSv3] POSIX 1003.1-2001 (ISO/IEC 9945-2003)
 [SUSv4] POSIX 1003.1-2008 (ISO/IEC 9945-2009)
 [SVID.4] SVID Issue 4

10.2.1 RPC

10.2.1.1 Interfaces for RPC

An LSB conforming implementation shall provide the architecture specific functions for RPC specified in Table 10-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-2 libc - RPC Function Interfaces

authnone_create(GLIBC_2.2.5) [SVID.4]	callrpc(GLIBC_2.2.5) [RPC + XDR]	clnt_create(GLIBC_2.2.5) [SVID.4]	clnt_pcreateerror(GLIBC_2.2.5) [SVID.4]
clnt_perrno(GLIBC_2.2.5) [SVID.4]	clnt_perror(GLIBC_2.2.5) [SVID.4]	clnt_spcreateerror(GLIBC_2.2.5) [SVID.4]	clnt_sperrno(GLIBC_2.2.5) [SVID.4]
clnt_sperror(GLIBC_2.2.5) [SVID.4]	clntraw_create(GLIBC_2.2.5) [RPC + XDR]	clnttcp_create(GLIBC_2.2.5) [RPC + XDR]	clntudp_bufcreate(GLIBC_2.2.5) [RPC + XDR]

clntudp_create(GLIBC_2.2.5) [RPC + XDR]	key_decryptsessi on(GLIBC_2.2.5) [SVID.4]	pmap_getport(GLIBC_2.2.5) [LSB]	pmap_set(GLIBC_2.2.5) [LSB]
pmap_unset(GLIBC_2.2.5) [LSB]	svc_getreqset(GLIBC_2.2.5) [SVID.4]	svc_register(GLIBC_2.2.5) [LSB]	svc_run(GLIBC_2.2.5) [LSB]
svc_sendreply(GLIBC_2.2.5) [LSB]	svcerr_auth(GLIBC_2.2.5) [SVID.4]	svcerr_decode(GLIBC_2.2.5) [SVID.4]	svcerr_noproc(GLIBC_2.2.5) [SVID.4]
svcerr_noprog(GLIBC_2.2.5) [SVID.4]	svcerr_progvers(GLIBC_2.2.5) [SVID.4]	svcerr_systemerr(GLIBC_2.2.5) [SVID.4]	svcerr_weakauth(GLIBC_2.2.5) [SVID.4]
svcfld_create(GLIBC_2.2.5) [RPC + XDR]	svccraw_create(GLIBC_2.2.5) [RPC + XDR]	svctcp_create(GLIBC_2.2.5) [LSB]	svcudp_create(GLIBC_2.2.5) [LSB]
xdr_accepted_reply(GLIBC_2.2.5) [SVID.4]	xdr_array(GLIBC_2.2.5) [SVID.4]	xdr_bool(GLIBC_2.2.5) [SVID.4]	xdr_bytes(GLIBC_2.2.5) [SVID.4]
xdr_callhdr(GLIBC_2.2.5) [SVID.4]	xdr_callmsg(GLIBC_2.2.5) [SVID.4]	xdr_char(GLIBC_2.2.5) [SVID.4]	xdr_double(GLIBC_2.2.5) [SVID.4]
xdr_enum(GLIBC_2.2.5) [SVID.4]	xdr_float(GLIBC_2.2.5) [SVID.4]	xdr_free(GLIBC_2.2.5) [SVID.4]	xdr_int(GLIBC_2.2.5) [SVID.4]
xdr_long(GLIBC_2.2.5) [SVID.4]	xdr_opaque(GLIBC_2.2.5) [SVID.4]	xdr_opaque_auth(GLIBC_2.2.5) [SVID.4]	xdr_pointer(GLIBC_2.2.5) [SVID.4]
xdr_reference(GLIBC_2.2.5) [SVID.4]	xdr_rejected_reply(GLIBC_2.2.5) [SVID.4]	xdr_replymsg(GLIBC_2.2.5) [SVID.4]	xdr_short(GLIBC_2.2.5) [SVID.4]
xdr_string(GLIBC_2.2.5) [SVID.4]	xdr_u_char(GLIBC_2.2.5) [SVID.4]	xdr_u_int(GLIBC_2.2.5) [LSB]	xdr_u_long(GLIBC_2.2.5) [SVID.4]
xdr_u_short(GLIBC_2.2.5) [SVID.4]	xdr_union(GLIBC_2.2.5) [SVID.4]	xdr_vector(GLIBC_2.2.5) [SVID.4]	xdr_void(GLIBC_2.2.5) [SVID.4]
xdr_wrapstring(GLIBC_2.2.5) [SVID.4]	xdrmem_create(GLIBC_2.2.5) [SVID.4]	xdrrec_create(GLIBC_2.2.5) [SVID.4]	xdrrec_endofrecord(GLIBC_2.2.5) [RPC + XDR]
xdrrec_eof(GLIBC_2.2.5) [SVID.4]	xdrrec_skiprecord(GLIBC_2.2.5) [RPC + XDR]	xdrstdio_create(GLIBC_2.2.5) [LSB]	

An LSB conforming implementation shall provide the architecture specific deprecated functions for RPC specified in Table 10-3, with the full mandatory functionality as described in the referenced underlying specification.

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

Table 10-3 libc - RPC Deprecated Function Interfaces

key_decryptsession(GLIBC_2.2.5) [SVID.4]			
--	--	--	--

10.2.2 Epoll

10.2.2.1 Interfaces for Epoll

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

10.2.3 System Calls

10.2.3.1 Interfaces for System Calls

An LSB conforming implementation shall provide the architecture specific functions for System Calls specified in Table 10-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-4 libc - System Calls Function Interfaces

__fxstat(GLIBC_2.2.5) [LSB]	__getpgid(GLIBC_2.2.5) [LSB]	__lxstat(GLIBC_2.2.5) [LSB]	__xmknod(GLIBC_2.2.5) [LSB]
__xstat(GLIBC_2.2.5) [LSB]	access(GLIBC_2.2.5) [SUSv4]	acct(GLIBC_2.2.5) [LSB]	alarm(GLIBC_2.2.5) [SUSv4]
backtrace(GLIBC_2.2.5) [LSB]	backtrace_symbols(GLIBC_2.2.5) [LSB]	backtrace_symbols_fd(GLIBC_2.2.5) [LSB]	brk(GLIBC_2.2.5) [SUSv2]
chdir(GLIBC_2.2.5) [SUSv4]	chmod(GLIBC_2.2.5) [SUSv4]	chown(GLIBC_2.2.5) [SUSv4]	chroot(GLIBC_2.2.5) [SUSv2]
clock(GLIBC_2.2.5) [SUSv4]	close(GLIBC_2.2.5) [SUSv4]	closedir(GLIBC_2.2.5) [SUSv4]	creat(GLIBC_2.2.5) [SUSv4]
dup(GLIBC_2.2.5) [SUSv4]	dup2(GLIBC_2.2.5) [SUSv4]	execl(GLIBC_2.2.5) [SUSv4]	execle(GLIBC_2.2.5) [SUSv4]
execlp(GLIBC_2.2.5) [SUSv4]	execv(GLIBC_2.2.5) [SUSv4]	execve(GLIBC_2.2.5) [SUSv4]	execvp(GLIBC_2.2.5) [SUSv4]
exit(GLIBC_2.2.5) [SUSv4]	fchdir(GLIBC_2.2.5) [SUSv4]	fchmod(GLIBC_2.2.5) [SUSv4]	fchown(GLIBC_2.2.5) [SUSv4]
fcntl(GLIBC_2.2.5) [LSB]	fdatasync(GLIBC_2.2.5) [SUSv4]	fexecve(GLIBC_2.2.5) [SUSv4]	flock(GLIBC_2.2.5) [LSB]
fork(GLIBC_2.2.5) [SUSv4]	fstatfs(GLIBC_2.2.5) [LSB]	fstatvfs(GLIBC_2.2.5) [SUSv4]	fsync(GLIBC_2.2.5) [SUSv4]
ftime(GLIBC_2.2.5) [SUSv3]	ftruncate(GLIBC_2.2.5) [SUSv4]	getcontext(GLIBC_2.2.5) [SUSv3]	getdtablesize(GLIBC_2.2.5) [LSB]

getegid(GLIBC_2.2.5) [SUSv4]	geteuid(GLIBC_2.2.5) [SUSv4]	getgid(GLIBC_2.2.5) [SUSv4]	getgroups(GLIBC_2.2.5) [SUSv4]
getitimer(GLIBC_2.2.5) [SUSv4]	getloadavg(GLIBC_2.2.5) [LSB]	getpagesize(GLIBC_2.2.5) [LSB]	getpgid(GLIBC_2.2.5) [SUSv4]
getpgrp(GLIBC_2.2.5) [SUSv4]	getpid(GLIBC_2.2.5) [SUSv4]	getppid(GLIBC_2.2.5) [SUSv4]	getpriority(GLIBC_2.2.5) [SUSv4]
getrlimit(GLIBC_2.2.5) [LSB]	getrusage(GLIBC_2.2.5) [SUSv4]	getsid(GLIBC_2.2.5) [SUSv4]	getuid(GLIBC_2.2.5) [SUSv4]
getwd(GLIBC_2.2.5) [SUSv3]	initgroups(GLIBC_2.2.5) [LSB]	ioctl(GLIBC_2.2.5) [LSB]	ioperm(GLIBC_2.2.5) [LSB]
iopl(GLIBC_2.2.5) [LSB]	kill(GLIBC_2.2.5) [LSB]	killpg(GLIBC_2.2.5) [SUSv4]	lchown(GLIBC_2.2.5) [SUSv4]
link(GLIBC_2.2.5) [LSB]	lockf(GLIBC_2.2.5) [SUSv4]	lseek(GLIBC_2.2.5) [SUSv4]	mkdir(GLIBC_2.2.5) [SUSv4]
mkfifo(GLIBC_2.2.5) [SUSv4]	mlock(GLIBC_2.2.5) [SUSv4]	mlockall(GLIBC_2.2.5) [SUSv4]	mmap(GLIBC_2.2.5) [SUSv4]
mprotect(GLIBC_2.2.5) [SUSv4]	mremap(GLIBC_2.2.5) [LSB]	msync(GLIBC_2.2.5) [SUSv4]	munlock(GLIBC_2.2.5) [SUSv4]
munlockall(GLIBC_2.2.5) [SUSv4]	munmap(GLIBC_2.2.5) [SUSv4]	nanosleep(GLIBC_2.2.5) [SUSv4]	nice(GLIBC_2.2.5) [SUSv4]
open(GLIBC_2.2.5) [SUSv4]	opendir(GLIBC_2.2.5) [SUSv4]	pathconf(GLIBC_2.2.5) [SUSv4]	pause(GLIBC_2.2.5) [SUSv4]
pipe(GLIBC_2.2.5) [SUSv4]	poll(GLIBC_2.2.5) [SUSv4]	pread(GLIBC_2.2.5) [SUSv4]	pselect(GLIBC_2.2.5) [SUSv4]
ptrace(GLIBC_2.2.5) [LSB]	pwrite(GLIBC_2.2.5) [SUSv4]	read(GLIBC_2.2.5) [SUSv4]	readdir(GLIBC_2.2.5) [SUSv4]
readdir_r(GLIBC_2.2.5) [SUSv4]	readlink(GLIBC_2.2.5) [SUSv4]	readv(GLIBC_2.2.5) [SUSv4]	rename(GLIBC_2.2.5) [SUSv4]
rmdir(GLIBC_2.2.5) [SUSv4]	sbrk(GLIBC_2.2.5) [SUSv2]	sched_get_priority_max(GLIBC_2.2.5) [SUSv4]	sched_get_priority_min(GLIBC_2.2.5) [SUSv4]
sched_getparam(GLIBC_2.2.5) [SUSv4]	sched_getscheduler(GLIBC_2.2.5) [SUSv4]	sched_rr_get_interval(GLIBC_2.2.5) [SUSv4]	sched_setparam(GLIBC_2.2.5) [SUSv4]
sched_setscheduler(GLIBC_2.2.5) [LSB]	sched_yield(GLIBC_2.2.5) [SUSv4]	select(GLIBC_2.2.5) [SUSv4]	setcontext(GLIBC_2.2.5) [SUSv3]
setegid(GLIBC_2.2.5) [SUSv4]	seteuid(GLIBC_2.2.5) [SUSv4]	setgid(GLIBC_2.2.5) [SUSv4]	setitimer(GLIBC_2.2.5) [SUSv4]
setpgid(GLIBC_2.2.5) [SUSv4]	setpgrp(GLIBC_2.2.5) [SUSv4]	setpriority(GLIBC_2.2.5) [SUSv4]	setregid(GLIBC_2.2.5) [SUSv4]