BS ISO/IEC 23360-3-2:2021

BSI Standards Publication

Linux Standard Base (LSB)

Part 3-2: Core specification for IA64 (Itanium™) architecture

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23360-3-2:2021

National foreword

This British Standard is the UK implementation of ISO/IEC 23360-3-2:2021.

The UK participation in its preparation was entrusted to Technical Committee IST/5, Programming languages, their environments and system software interfaces.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2022
Published by BSI Standards Limited 2022

ISBN 978 0 539 05256 5

ICS 35.080

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2022.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23360-3-2:2021

INTERNATIONAL STANDARD

ISO/IEC 23360-3-2

First edition
2021-10

Linux Standard Base (LSB) —

Part 3-2:
Core specification for IA64 (Itanium™) architecture

Reference number
ISO/IEC 23360-3-2:2021(E)

Documents
Document Sharing Hub
© ISO/IEC 2021

Full document details: BS ISO/IEC 23360-3-2:2021

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

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
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 23360-3-2:2021

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the Linux Foundation as Linux Standard Base (LSB): Core specification for IA64 (Itanium ™) architecture and drafted in accordance with its editorial rules. It was assigned to Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*, and adopted by National Bodies.

This first edition of ISO/IEC 23360-3-2 cancels and replaces ISO/IEC 23360-3:2006, which has been technically revised.

This document is based on “The GNU Free Documentation License, version 1.1”. The license is available at https://www.gnu.org/licenses/old-licenses/fdl-1.1.html.

A list of all parts in the ISO/IEC 23660 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

© ISO/IEC 2021 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Contents

Foreword ... **iii**
Introduction ... **vi**
I Introductory Elements ... **1**
1 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 ... 20
7.3 Operating System Interface ... 21
7.4 Process Initialization ... 22
7.5 Coding Examples ... 24
7.6 C Stack Frame ... 25
7.7 Debug Information ... 26
8 Object Format ... 27
8.1 Introduction ... 27
8.2 ELF Header ... 27
8.3 Sections ... 28
8.4 Symbol Table ... 30
8.5 Relocation ... 30
9 Program Loading and Dynamic Linking ... 31
9.1 Introduction ... 31
9.2 Program Header ... 31
9.3 Program Loading ... 31
9.4 Dynamic Linking ... 31
III Base Libraries ... **33**
10 Libraries ... 34
10.1 Program Interpreter/Dynamic Linker ... 34
10.2 Interfaces for libc ... 34
10.3 Data Definitions for libc ... 54
10.4 Interface Definitions for libc ... 72
10.5 Interfaces for libm ... 74
10.6 Data Definitions for libm ... 79
10.7 Interface Definitions for libm ... 80
10.8 Interfaces for libpthread ... 81
10.9 Data Definitions for libpthread ... 86
10.10 Interfaces for libgcc_s ... 88
10.11 Data Definitions for libgcc_s ... 88
10.12 Interface Definitions for libgcc_s ... 89
10.13 Interfaces for libdl ... 89
10.14 Data Definitions for libdl ... 90
10.15 Interfaces for libcrypt ... 90
10.16 Data Definitions for libcrypt ... 91
IV Utility Libraries ... **92**
11 Libraries ... 93

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

11.1 Interfaces for libz 93
11.2 Data Definitions for libz 93
11.3 Interfaces for libncurses 94
11.4 Data Definitions for libncurses 94
11.5 Interfaces for libncursesw 94
11.6 Data Definitions for libncursesw 95
11.7 Interfaces for libutil 95
V Base Libraries 97
12 Libraries 98
12.1 Interfaces for libstdcxx 98
12.2 Interface Definitions for libstdcxx 209
VI Package Format and Installation 210
13 Software Installation 211
13.1 Package Dependencies 211
13.2 Package Architecture Considerations 211
Annex A Alphabetical Listing of Interfaces by Library 212
A.1 libc 212
A.2 libcrypt 227
A.3 libdl 227
A.4 libgcc_s 228
A.5 libm 228
A.6 libpthread 233
A.7 librt 236
A.8 libutil 237

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

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.

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

I Introductory Elements

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

1 Scope

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.

This is the Itanium™ 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.

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

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/
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 3.0	http://refspecs.linuxbase.org/fhs
Intel® Itanium™ Processor-specific Application Binary Interface	Intel® Itanium™ Processor-specific Application Binary Interface	http://refspecs.linux-foundation.org/elf/IA64-SysV-psABI.pdf
ISO C (1999)	ISO/IEC 9899:1999 - Programming Languages -- C	
ISO/IEC 14882: 2003 C++ Language	ISO/IEC 14882: 2003 Programming languages --C++	
Itanium™ Architecture Software Developer's Manual Volume 1	Itanium™ Architecture Software Developer's Manual Volume 1: Application Architecture	http://refspecs.linux-foundation.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.linux-foundation.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.linux-foundation.org/IA64-softdevman-vol3.pdf
Itanium™ Architecture Software Developer's Manual Volume 4	IA-64 Processor Reference: Intel® Itanium™ Processor	http://refspecs.linux-foundation.org/IA64-softdevman-vol4.pdf

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Name	Title	URL
	Reference Manual for Software Development	
Itanium™ C++ ABI	Itanium™ C++ ABI (Revision 1.86)	http://refspecs.linuxfoundation.org/cxxabi-1.86.html
Itanium™ Software Conventions and Runtime Guide	Itanium™ Software Conventions & Runtime Architecture Guide, September 2000	http://refspecs.linux-foundation.org/IA64conventions.pdf
Large File Support	Large File Support	http://www.UNIX-systems.org/version2/whatsnew/lfs20mar.html
Libncursesw API	Libncursesw API	http://invisible-island.net/ncurses/man/ncurses.3x.html
Libncursesw Placeholder	Libncursesw Specification Placeholder	http://refspecs.linux-foundation.org/libncursesw/libncurses.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/

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

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
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

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Name	Title	URL
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 conversionITUV	http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-V.42
Li18nux Globalization Specification	LI18NUX 2000 Globalization Specification, Version 1.0 with Amendment 4	http://www.openi18n.org/docs/html/LI18NUX-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
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

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

Name	Title	URL
RFC 1950: ZLIB Compressed Data Format Specication	IETF RFC 1950: ZLIB Compressed Data Format Specification	http://www.ietf.org/rfc/rfc1950.txt
RFC 1951: DEFLATE Compressed Data Format Specification	IETF RFC 1951: DEFLATE Compressed Data Format Specification version 1.3	http://www.ietf.org/rfc/rfc1951.txt
RFC 1952: GZIP File Format Specification	IETF RFC 1952: GZIP file format specification version 4.3	http://www.ietf.org/rfc/rfc1952.txt
RFC 2440: OpenPGP Message Format	IETF RFC 2440: OpenPGP Message Format	http://www.ietf.org/rfc/rfc2440.txt
RFC 2821:Simple Mail Transfer Protocol	IETF RFC 2821: Simple Mail Transfer Protocol	http://www.ietf.org/rfc/rfc2821.txt
RFC 2822:Internet Message Format	IETF RFC 2822: Internet Message Format	http://www.ietf.org/rfc/rfc2822.txt
RFC 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/

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on IA64 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.6.1
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libgcc_s	libgcc_s.so.1
libm	libm.so.6.1
libncurses	libncurses.so.5
libncursesw	libncursesw.so.5
libpthread	libpthread.so.0
libstdcxx	libstdc++.so.6
libutil	libutil.so.1
libz	libz.so.1
proginterp	/lib/ld-lsb-ia64.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.

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

- 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

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

- 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.

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

4 Terms and Definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 2382, ISO 80000–2, and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

4.1

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.

4.2

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.

4.3

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.

4.4

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.

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

4.5

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.

4.6

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.

4.7

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.

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

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.

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

II Executable and Linking Format (ELF)

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

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 Intel® Itanium™ Processor-specific Application Binary Interface, and is intended to document additions made since the publication of that document.

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

7 Low Level System Information

7.1 Machine Interface

7.1.1 Processor Architecture

The Itanium™ Architecture is specified by the following documents

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

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

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.

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

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

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

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

7.1.2 Data Representation

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

Within this specification, the term `byte` refers to an 8-bit object, the term `halfword` refers to a 16-bit object, the term `word` refers to a 32-bit object, the term

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

`doubleword` refers to a 64-bit object, and the term `quadword` refers to a 128-bit object.

7.1.2.1 Byte Ordering

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

7.1.2.2 Fundamental Types

Table 7-1 describes how fundemental C language data types shall be represented:

Table 7-1 Scalar Types

Type	C	`sizeof`	Alignment (bytes)	Hardware Representation
Integral	_Bool	1	1	byte (sign unspecified)
	char	1	1	signed byte
	signed char			
	unsigned char	2	2	signed byte
	short			signed half-word
	signed short			
	unsigned short	4	4	unsigned halfword
	int			signed word
	signed int			
	unsigned int	8	8	unsigned word
	long			signed double-word
	signed long			
	unsigned long	8	8	unsigned doubleword
	long long			signed double-word
	signed long long			
	unsigned long long	8	8	unsigned doubleword
Pointer	*any-type* *	8	8	unsigned doubleword

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Type	C	sizeof	Alignment (bytes)	Hardware Representation
	any-type (*) ()			
Floating-Point	float	4	4	IEEE Single-precision
	double	8	8	IEEE Double-precision
	long double	16	16	IEEE Double-extended

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

7.1.2.3 Aggregates and Unions

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

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

A conforming application shall not read padding.

```
struct {
    char c;
}
```

Byte aligned, `sizeof` is 1

Offset	Byte 0
0	c^0

Figure 7-1 Structure Smaller Than A Word

```
struct {
    char  c;
    char  d;
    short s;
    int   i;
    long  l;
}
```

Doubleword Aligned, `sizeof` is 16

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Offset	Byte 3	Byte 2	Byte 1	Byte 0
0	s^2		d^1	c^0
4	i^0			
8	l^0			
12				

Figure 7-2 No Padding

```
struct {
    char  c;
    long  l;
    int   i;
    short s;
}
```

Doubleword Aligned, `sizeof` is 24

Offset	Byte 3	Byte 2	Byte 1	Byte 0
0	pad^1			c^0
4	pad^1			
8	l^0			
12				
16	i^0			
20	pad^2		s^0	

Figure 7-3 Internal and Tail Padding

7.1.2.4 Bit Fields

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

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

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

Bit-field Type	Width *w*	Range
`signed char`	1 to 8	-2^{w-1} to $2^{w-1}-1$

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Bit-field Type	Width *w*	Range
`char` `unsigned char`		0 to $2^{w}-1$ 0 to $2^{w}-1$

`signed short`
`short`
`unsigned short`
	1 to 16	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

`signed int`
`int`
`unsigned int`
	1 to 32	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

`signed long`
`long`
`unsigned long`
	1 to 64	-2^{w-1} to $2^{w-1}-1$
		0 to $2^{w}-1$
		0 to $2^{w}-1$

Figure 7-4 Bit-Field Ranges

7.2 Function Calling Sequence

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

7.2.1 Registers

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

7.2.2 Floating Point Registers

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

7.2.3 Stack Frame

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

7.2.4 Arguments

7.2.4.1 Introduction

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

7.2.4.2 Integral/Pointer

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

7.2.4.3 Floating Point

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

7.2.4.4 Struct and Union Point

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.

7.2.4.5 Variable Arguments

See Itanium™ Software Conventions and Runtime Guide Chapter 8.5.4.

7.2.5 Return Values

7.2.5.1 Introduction

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

7.2.5.2 Void

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

7.2.5.3 Integral/Pointer

See Itanium™ Software Conventions and Runtime Guide Chapter 8.6.

7.2.5.4 Floating Point

See Itanium™ Software Conventions and Runtime Guide Chapter 8.6.

7.2.5.5 Struct and Union

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

7.3 Operating System Interface

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

7.3.1 Processor Execution Mode

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

7.3.2 Exception Interface

7.3.2.1 Introduction

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

7.3.2.2 Hardware Exception Types

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

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

7.3.2.3 Software Trap Types

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

7.3.3 Signal Delivery

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

7.3.3.1 Signal Handler Interface

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

7.3.4 Debugging Support

The LSB does not specify debugging information.

7.3.5 Process Startup

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

7.4 Process Initialization

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

7.4.1 Special Registers

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

7.4.2 Process Stack (on entry)

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

7.4.3 Auxiliary Vector

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

```
typedef struct
{
  long int a_type;            /* Entry type */
  union
    {
      long int a_val;         /* Integer value */
      void *a_ptr;           /* Pointer value */
      void (*a_fcn) (void);    /* Function pointer value */
    } a_un;
} auxv_t;
```


© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

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

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

AT_NULL

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

AT_IGNORE

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

AT_EXECFD

File descriptor of program

AT_PHDR

Program headers for program

AT_PHENT

Size of program header entry

AT_PHNUM

Number of program headers

AT_PAGESZ

System page size

AT_BASE

Base address of interpreter

AT_FLAGS

Flags

AT_ENTRY

Entry point of program

AT_NOTELF

Program is not ELF

AT_UID

Real uid

AT_EUID

Effective uid

AT_GID

Real gid

AT_EGID

Effective gid

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

AT_CLKTCK

Frequency of times()

AT_PLATFORM

String identifying platform.

AT_HWCAP

Machine dependent hints about processor capabilities.

AT_FPUCW

Used FPU control word

AT_DCACHEBSIZE

Data cache block size

AT_ICACHEBSIZE

Instruction cache block size

AT_UCACHEBSIZE

Unified cache block size

Note: The auxiliary vector is intended for passing information from the operating system to the program interpreter.

7.4.4 Environment

Although a pointer to the environment vector should be available as a third argument to the `main()` entry point, conforming applications should use `getenv()` to access the environment. (See POSIX 1003.1-2008 (ISO/IEC 9945-2009), Section `exec()`).

7.5 Coding Examples

7.5.1 Introduction

LSB-conforming applications may implement fundamental operations using the Coding Examples as shown below.

Sample code sequences and coding conventions can be found in Itanium™ Software Conventions and Runtime Guide, Chapter 9.

7.5.2 Code Model Overview/Architecture Constraints

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

7.5.3 Position-Independent Function Prologue

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

7.5.4 Data Objects

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

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

7.5.4.1 Absolute Load & Store

Conforming applications shall not use absolute addressing.

7.5.4.2 Position Relative Load & Store

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

7.5.5 Function Calls

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

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

7.5.5.1 Absolute Direct Function Call

Conforming applications shall not use absolute addressing.

7.5.5.2 Absolute Indirect Function Call

Conforming applications shall not use absolute addressing.

7.5.5.3 Position-Independent Direct Function Call

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

7.5.5.4 Position-Independent Indirect Function Call

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

7.5.6 Branching

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

7.5.6.1 Branch Instruction

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

7.5.6.2 Absolute switch() code

Conforming applications shall not use absolute addressing.

7.5.6.3 Position-Independent switch() code

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

7.6 C Stack Frame

7.6.1 Variable Argument List

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

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

7.6.2 Dynamic Allocation of Stack Space

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

7.7 Debug Information

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

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

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
- Intel® Itanium™ Processor-specific Application Binary Interface
- LSB Core - Generic
- this document

8.2 ELF Header

8.2.1 Machine Information

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

8.2.1.1 File Class

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

8.2.1.2 Data Encoding

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

8.2.1.3 OS Identification

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

8.2.1.4 Processor Identification

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

8.2.1.5 Processor Specific Flags

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

The following additional processor-specific flags are defined:

Table 8-1 Additional Processor-Specific Flags

Name	Value
EF_IA_64_LINUX_EXECUTABLE_STACK	0x00000001

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

EF_IA_64_LINUX_EXECUTABLE_STACK

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

8.3 Sections

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

8.3.1 Special Sections

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

Table 8-2 ELF Special Sections

Name	Type	Attributes
.got	SHT_PROGBITS	SHF_AL-LOC+SHF_WRITE+SHF_IA_64_SHORT
.IA_64.archext	SHT_IA_64_EXT	0
.IA_64.pltoff	SHT_PROGBITS	SHF_AL-LOC+SHF_WRITE+SHF_IA_64_SHORT
.IA_64.unwind	SHT_IA_64_UNWIND	SHF_AL-LOC+SHF_LINK_OR-DER
.IA_64.unwind_info	SHT_PROGBITS	SHF_ALLOC
.plt	SHT_PROGBITS	SHF_ALLOC+SHF_EX-ECINSTR
.sbss	SHT_NOBITS	SHF_AL-LOC+SHF_WRITE+SHF_IA_64_SHORT
.sdata	SHT_PROGBITS	SHF_AL-LOC+SHF_WRITE+SHF_IA_64_SHORT
.sdata1	SHT_PROGBITS	SHF_AL-LOC+SHF_WRITE+SHF_IA_64_SHORT

.got

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

.IA_64.archext

This section holds product-specific extension bits. The link editor will perform a logical "or" of the extension bits of each object when creating an

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

executable so that it creates only a single .IA_64.archext section in the executable.

.IA_64.pltoff

This section holds local function descriptor entries.

.IA_64.unwind

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

.IA_64.unwind_info

This section holds stack unwind and and exception handling information. The exception handling information is programming language specific, and is unspecified.

.plt

This section holds the procedure linkage table.

.sbss

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

.sdata

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

.sdata1

See .sdata.

8.3.2 Linux Special Sections

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

Table 8-3 Additional Special Sections

Name	Type	Attributes
.opd	SHT_PROGBITS	SHF_ALLOC
.rela.dyn	SHT_RELA	SHF_ALLOC
.rela.IA_64.pltoff	SHT_RELA	SHF_ALLOC

.opd

This section holds function descriptors.

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

.rela.dyn

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

.rela.IA_64.pltoff

This section holds relocation information, as described in `Relocation' section in Chapter 4 of System V ABI Update. These relocations are applied to the .IA_64.pltoff section.

8.3.3 Section Types

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

8.3.4 Section Attribute Flags

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

8.3.5 Special Section Types

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

8.4 Symbol Table

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

8.5 Relocation

8.5.1 Relocation Types

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

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

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, Intel® Itanium™ Processor-specific Application Binary Interface and as supplemented by the Linux Standard Base Specification and this document.

9.2 Program Header

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

9.2.1 Types

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

9.2.2 Flags

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

9.3 Program Loading

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

9.4 Dynamic Linking

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

9.4.1 Dynamic Entries

9.4.1.1 ELF Dynamic Entries

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

DT_PLTGOT

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

9.4.1.2 Additional Dynamic Entries

The following dynamic entries are defined here.

DT_RELACOUNT

The number of relative relocations in .rela.dyn

9.4.2 Global Offset Table

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

9.4.3 Shared Object Dependencies

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

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

9.4.4 Function Addresses

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

9.4.5 Procedure Linkage Table

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

9.4.6 Initialization and Termination Functions

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

Documents
Document Sharing Hub

© 2021 ISO/IEC – All rights reserved

Full document details: BS ISO/IEC 23360-3-2:2021

III Base Libraries

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

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 Itanium™ 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 `/lib/ld-lsb-ia64.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.6.1

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) [SVID.4]	callrpc(GLIBC_2.2) [RPC + XDR]	clnt_create(GLIBC_2.2) [SVID.4]	clnt_pcreateerror(GLIBC_2.2) [SVID.4]
clnt_perrno(GLIBC_2.2) [SVID.4]	clnt_perror(GLIBC_2.2) [SVID.4]	clnt_spcreateerror(GLIBC_2.2) [SVID.4]	clnt_sperrno(GLIBC_2.2) [SVID.4]
clnt_sperror(GLIBC_2.2) [SVID.4]	clntraw_create(GLIBC_2.2) [RPC + XDR]	clnttcp_create(GLIBC_2.2) [RPC + XDR]	clntudp_bufcreate(GLIBC_2.2) [RPC + XDR]

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

clntudp_create(GLIBC_2.2) [RPC + XDR]	key_decryptsession(GLIBC_2.2) [SVID.4]	pmap_getport(GLIBC_2.2) [LSB]	pmap_set(GLIBC_2.2) [LSB]
pmap_unset(GLIBC_2.2) [LSB]	svc_getreqset(GLIBC_2.2) [SVID.4]	svc_register(GLIBC_2.2) [LSB]	svc_run(GLIBC_2.2) [LSB]
svc_sendreply(GLIBC_2.2) [LSB]	svcerr_auth(GLIBC_2.2) [SVID.4]	svcerr_decode(GLIBC_2.2) [SVID.4]	svcerr_noproc(GLIBC_2.2) [SVID.4]
svcerr_noprog(GLIBC_2.2) [SVID.4]	svcerr_progvers(GLIBC_2.2) [SVID.4]	svcerr_systemerr(GLIBC_2.2) [SVID.4]	svcerr_weakauth(GLIBC_2.2) [SVID.4]
svcfd_create(GLIBC_2.2) [RPC + XDR]	svcraw_create(GLIBC_2.2) [RPC + XDR]	svctcp_create(GLIBC_2.2) [LSB]	svcudp_create(GLIBC_2.2) [LSB]
xdr_accepted_reply(GLIBC_2.2) [SVID.4]	xdr_array(GLIBC_2.2) [SVID.4]	xdr_bool(GLIBC_2.2) [SVID.4]	xdr_bytes(GLIBC_2.2) [SVID.4]
xdr_callhdr(GLIBC_2.2) [SVID.4]	xdr_callmsg(GLIBC_2.2) [SVID.4]	xdr_char(GLIBC_2.2) [SVID.4]	xdr_double(GLIBC_2.2) [SVID.4]
xdr_enum(GLIBC_2.2) [SVID.4]	xdr_float(GLIBC_2.2) [SVID.4]	xdr_free(GLIBC_2.2) [SVID.4]	xdr_int(GLIBC_2.2) [SVID.4]
xdr_long(GLIBC_2.2) [SVID.4]	xdr_opaque(GLIBC_2.2) [SVID.4]	xdr_opaque_auth(GLIBC_2.2) [SVID.4]	xdr_pointer(GLIBC_2.2) [SVID.4]
xdr_reference(GLIBC_2.2) [SVID.4]	xdr_rejected_reply(GLIBC_2.2) [SVID.4]	xdr_replymsg(GLIBC_2.2) [SVID.4]	xdr_short(GLIBC_2.2) [SVID.4]
xdr_string(GLIBC_2.2) [SVID.4]	xdr_u_char(GLIBC_2.2) [SVID.4]	xdr_u_int(GLIBC_2.2) [LSB]	xdr_u_long(GLIBC_2.2) [SVID.4]
xdr_u_short(GLIBC_2.2) [SVID.4]	xdr_union(GLIBC_2.2) [SVID.4]	xdr_vector(GLIBC_2.2) [SVID.4]	xdr_void(GLIBC_2.2) [SVID.4]
xdr_wrapstring(GLIBC_2.2) [SVID.4]	xdrmem_create(GLIBC_2.2) [SVID.4]	xdrrec_create(GLIBC_2.2) [SVID.4]	xdrrec_endofrecord(GLIBC_2.2) [RPC + XDR]
xdrrec_eof(GLIBC_2.2) [SVID.4]	xdrrec_skiprecord(GLIBC_2.2) [RPC + XDR]	xdrstdio_create(GLIBC_2.2) [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.

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Table 10-3 libc - RPC Deprecated Function Interfaces

key_decryptsession(GLIBC_2.2) [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) [LSB]	__getpgid(GLIBC_2.2) [LSB]	__lxstat(GLIBC_2.2) [LSB]	__xmknod(GLIBC_2.2) [LSB]
__xstat(GLIBC_2.2) [LSB]	access(GLIBC_2.2) [SUSv4]	acct(GLIBC_2.2) [LSB]	alarm(GLIBC_2.2) [SUSv4]
backtrace(GLIBC_2.2) [LSB]	backtrace_symbols(GLIBC_2.2) [LSB]	backtrace_symbols_fd(GLIBC_2.2) [LSB]	brk(GLIBC_2.2) [SUSv2]
chdir(GLIBC_2.2) [SUSv4]	chmod(GLIBC_2.2) [SUSv4]	chown(GLIBC_2.2) [SUSv4]	chroot(GLIBC_2.2) [SUSv2]
clock(GLIBC_2.2) [SUSv4]	close(GLIBC_2.2) [SUSv4]	closedir(GLIBC_2.2) [SUSv4]	creat(GLIBC_2.2) [SUSv4]
dup(GLIBC_2.2) [SUSv4]	dup2(GLIBC_2.2) [SUSv4]	execl(GLIBC_2.2) [SUSv4]	execle(GLIBC_2.2) [SUSv4]
execlp(GLIBC_2.2) [SUSv4]	execv(GLIBC_2.2) [SUSv4]	execve(GLIBC_2.2) [SUSv4]	execvp(GLIBC_2.2) [SUSv4]
exit(GLIBC_2.2) [SUSv4]	fchdir(GLIBC_2.2) [SUSv4]	fchmod(GLIBC_2.2) [SUSv4]	fchown(GLIBC_2.2) [SUSv4]
fcntl(GLIBC_2.2) [LSB]	fdatasync(GLIBC_2.2) [SUSv4]	fexecve(GLIBC_2.2) [SUSv4]	flock(GLIBC_2.2) [LSB]
fork(GLIBC_2.2) [SUSv4]	fstatfs(GLIBC_2.2) [LSB]	fstatvfs(GLIBC_2.2) [SUSv4]	fsync(GLIBC_2.2) [SUSv4]
ftime(GLIBC_2.2) [SUSv3]	ftruncate(GLIBC_2.2) [SUSv4]	getcontext(GLIBC_2.2) [SUSv3]	getdtablesize(GLIBC_2.2) [LSB]
getegid(GLIBC_2.2) [SUSv4]	geteuid(GLIBC_2.2) [SUSv4]	getgid(GLIBC_2.2) [SUSv4]	getgroups(GLIBC_2.2) [SUSv4]

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

getitimer(GLIBC_2.2) [SUSv4]	getloadavg(GLIBC_2.2) [LSB]	getpagesize(GLIBC_2.2) [LSB]	getpgid(GLIBC_2.2) [SUSv4]
getpgrp(GLIBC_2.2) [SUSv4]	getpid(GLIBC_2.2) [SUSv4]	getppid(GLIBC_2.2) [SUSv4]	getpriority(GLIBC_2.2) [SUSv4]
getrlimit(GLIBC_2.2) [LSB]	getrusage(GLIBC_2.2) [SUSv4]	getsid(GLIBC_2.2) [SUSv4]	getuid(GLIBC_2.2) [SUSv4]
getwd(GLIBC_2.2) [SUSv3]	initgroups(GLIBC_2.2) [LSB]	ioctl(GLIBC_2.2) [LSB]	ioperm(GLIBC_2.2) [LSB]
iopl(GLIBC_2.2) [LSB]	kill(GLIBC_2.2) [LSB]	killpg(GLIBC_2.2) [SUSv4]	lchown(GLIBC_2.2) [SUSv4]
link(GLIBC_2.2) [LSB]	lockf(GLIBC_2.2) [SUSv4]	lseek(GLIBC_2.2) [SUSv4]	mkdir(GLIBC_2.2) [SUSv4]
mkfifo(GLIBC_2.2) [SUSv4]	mlock(GLIBC_2.2) [SUSv4]	mlockall(GLIBC_2.2) [SUSv4]	mmap(GLIBC_2.2) [SUSv4]
mprotect(GLIBC_2.2) [SUSv4]	mremap(GLIBC_2.2) [LSB]	msync(GLIBC_2.2) [SUSv4]	munlock(GLIBC_2.2) [SUSv4]
munlockall(GLIBC_2.2) [SUSv4]	munmap(GLIBC_2.2) [SUSv4]	nanosleep(GLIBC_2.2) [SUSv4]	nice(GLIBC_2.2) [SUSv4]
open(GLIBC_2.2) [SUSv4]	opendir(GLIBC_2.2) [SUSv4]	pathconf(GLIBC_2.2) [SUSv4]	pause(GLIBC_2.2) [SUSv4]
pipe(GLIBC_2.2) [SUSv4]	poll(GLIBC_2.2) [SUSv4]	pread(GLIBC_2.2) [SUSv4]	pselect(GLIBC_2.2) [SUSv4]
ptrace(GLIBC_2.2) [LSB]	pwrite(GLIBC_2.2) [SUSv4]	read(GLIBC_2.2) [SUSv4]	readdir(GLIBC_2.2) [SUSv4]
readdir_r(GLIBC_2.2) [SUSv4]	readlink(GLIBC_2.2) [SUSv4]	readv(GLIBC_2.2) [SUSv4]	rename(GLIBC_2.2) [SUSv4]
rmdir(GLIBC_2.2) [SUSv4]	sbrk(GLIBC_2.2) [SUSv2]	sched_get_priority_max(GLIBC_2.2) [SUSv4]	sched_get_priority_min(GLIBC_2.2) [SUSv4]
sched_getparam(GLIBC_2.2) [SUSv4]	sched_getscheduler(GLIBC_2.2) [SUSv4]	sched_rr_get_interval(GLIBC_2.2) [SUSv4]	sched_setparam(GLIBC_2.2) [SUSv4]
sched_setscheduler(GLIBC_2.2) [LSB]	sched_yield(GLIBC_2.2) [SUSv4]	select(GLIBC_2.2) [SUSv4]	setcontext(GLIBC_2.2) [SUSv3]
setegid(GLIBC_2.2) [SUSv4]	seteuid(GLIBC_2.2) [SUSv4]	setgid(GLIBC_2.2) [SUSv4]	setitimer(GLIBC_2.2) [SUSv4]
setpgid(GLIBC_2.2) [SUSv4]	setpgrp(GLIBC_2.2) [SUSv4]	setpriority(GLIBC_2.2) [SUSv4]	setregid(GLIBC_2.2) [SUSv4]
setreuid(GLIBC_2.2) [SUSv4]	setrlimit(GLIBC_2.2) [LSB]	setrlimit64(GLIBC_2.2) [LFS]	setsid(GLIBC_2.2) [SUSv4]
setuid(GLIBC_2.2) [SUSv4]	sleep(GLIBC_2.2) [SUSv4]	statfs(GLIBC_2.2) [LSB]	statvfs(GLIBC_2.2) [SUSv4]

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

stime(GLIBC_2.2) [LSB]	symlink(GLIBC_2.2) [SUSv4]	sync(GLIBC_2.2) [SUSv4]	sysconf(GLIBC_2.2) [LSB]
sysinfo(GLIBC_2.2) [LSB]	time(GLIBC_2.2) [SUSv4]	times(GLIBC_2.2) [SUSv4]	truncate(GLIBC_2.2) [SUSv4]
ulimit(GLIBC_2.2) [SUSv4]	umask(GLIBC_2.2) [SUSv4]	uname(GLIBC_2.2) [SUSv4]	unlink(GLIBC_2.2) [LSB]
utime(GLIBC_2.2) [SUSv4]	utimes(GLIBC_2.2) [SUSv4]	vfork(GLIBC_2.2) [SUSv3]	wait(GLIBC_2.2) [SUSv4]
wait4(GLIBC_2.2) [LSB]	waitid(GLIBC_2.2) [SUSv4]	waitpid(GLIBC_2.2) [SUSv4]	write(GLIBC_2.2) [SUSv4]
writev(GLIBC_2.2) [SUSv4]			

An LSB conforming implementation shall provide the architecture specific deprecated functions for System Calls specified in Table 10-5, 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 10-5 libc - System Calls Deprecated Function Interfaces

fstatfs(GLIBC_2.2) [LSB]	getdtablesize(GLIBC_2.2) [LSB]	getpagesize(GLIBC_2.2) [LSB]	getwd(GLIBC_2.2) [SUSv3]
statfs(GLIBC_2.2) [LSB]			

10.2.4 Standard I/O

10.2.4.1 Interfaces for Standard I/O

An LSB conforming implementation shall provide the architecture specific functions for Standard I/O specified in Table 10-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-6 libc - Standard I/O Function Interfaces

_IO_feof(GLIBC_2.2) [LSB]	_IO_getc(GLIBC_2.2) [LSB]	_IO_putc(GLIBC_2.2) [LSB]	_IO_puts(GLIBC_2.2) [LSB]
__fprintf_chk(GLIBC_2.3.4) [LSB]	__printf_chk(GLIBC_2.3.4) [LSB]	__snprintf_chk(GLIBC_2.3.4) [LSB]	__sprintf_chk(GLIBC_2.3.4) [LSB]
__vfprintf_chk(GLIBC_2.3.4) [LSB]	__vprintf_chk(GLIBC_2.3.4) [LSB]	__vsnprintf_chk(GLIBC_2.3.4) [LSB]	__vsprintf_chk(GLIBC_2.3.4) [LSB]
asprintf(GLIBC_2.2) [LSB]	clearerr(GLIBC_2.2) [SUSv4]	clearerr_unlocked(GLIBC_2.2) [LSB]	ctermid(GLIBC_2.2) [SUSv4]
dprintf(GLIBC_2.2) [SUSv4]	fclose(GLIBC_2.2) [SUSv4]	fdopen(GLIBC_2.2) [SUSv4]	feof(GLIBC_2.2) [SUSv4]

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

feof_unlocked(GLIBC_2.2) [LSB]	ferror(GLIBC_2.2) [SUSv4]	ferror_unlocked(GLIBC_2.2) [LSB]	fflush(GLIBC_2.2) [SUSv4]
fflush_unlocked(GLIBC_2.2) [LSB]	fgetc(GLIBC_2.2) [SUSv4]	fgetc_unlocked(GLIBC_2.2) [LSB]	fgetpos(GLIBC_2.2) [SUSv4]
fgets(GLIBC_2.2) [SUSv4]	fgets_unlocked(GLIBC_2.2) [LSB]	fgetwc_unlocked (GLIBC_2.2) [LSB]	fgetws_unlocked (GLIBC_2.2) [LSB]
fileno(GLIBC_2.2) [SUSv4]	fileno_unlocked(GLIBC_2.2) [LSB]	flockfile(GLIBC_2.2) [SUSv4]	fopen(GLIBC_2.2) [SUSv4]
fprintf(GLIBC_2.2) [SUSv4]	fputc(GLIBC_2.2) [SUSv4]	fputc_unlocked(GLIBC_2.2) [LSB]	fputs(GLIBC_2.2) [SUSv4]
fputs_unlocked(GLIBC_2.2) [LSB]	fputwc_unlocked (GLIBC_2.2) [LSB]	fputws_unlocked (GLIBC_2.2) [LSB]	fread(GLIBC_2.2) [SUSv4]
fread_unlocked(GLIBC_2.2) [LSB]	freopen(GLIBC_2.2) [SUSv4]	fscanf(GLIBC_2.2) [LSB]	fseek(GLIBC_2.2) [SUSv4]
fseeko(GLIBC_2.2) [SUSv4]	fsetpos(GLIBC_2.2) [SUSv4]	ftell(GLIBC_2.2) [SUSv4]	ftello(GLIBC_2.2) [SUSv4]
fwrite(GLIBC_2.2) [SUSv4]	fwrite_unlocked(GLIBC_2.2) [LSB]	getc(GLIBC_2.2) [SUSv4]	getc_unlocked(GLIBC_2.2) [SUSv4]
getchar(GLIBC_2.2) [SUSv4]	getchar_unlocked(GLIBC_2.2) [SUSv4]	getdelim(GLIBC_2.2) [SUSv4]	getline(GLIBC_2.2) [SUSv4]
getw(GLIBC_2.2) [SUSv2]	getwc_unlocked(GLIBC_2.2) [LSB]	getwchar_unlocked(GLIBC_2.2) [LSB]	pclose(GLIBC_2.2) [SUSv4]
popen(GLIBC_2.2) [SUSv4]	printf(GLIBC_2.2) [SUSv4]	putc(GLIBC_2.2) [SUSv4]	putc_unlocked(GLIBC_2.2) [SUSv4]
putchar(GLIBC_2.2) [SUSv4]	putchar_unlocked(GLIBC_2.2) [SUSv4]	puts(GLIBC_2.2) [SUSv4]	putw(GLIBC_2.2) [SUSv2]
putwc_unlocked(GLIBC_2.2) [LSB]	putwchar_unlocked(GLIBC_2.2) [LSB]	remove(GLIBC_2.2) [SUSv4]	rewind(GLIBC_2.2) [SUSv4]
rewinddir(GLIBC_2.2) [SUSv4]	scanf(GLIBC_2.2) [LSB]	seekdir(GLIBC_2.2) [SUSv4]	setbuf(GLIBC_2.2) [SUSv4]
setbuffer(GLIBC_2.2) [LSB]	setvbuf(GLIBC_2.2) [SUSv4]	snprintf(GLIBC_2.2) [SUSv4]	sprintf(GLIBC_2.2) [SUSv4]
sscanf(GLIBC_2.2) [LSB]	telldir(GLIBC_2.2) [SUSv4]	tempnam(GLIBC_2.2) [SUSv4]	ungetc(GLIBC_2.2) [SUSv4]
vasprintf(GLIBC_2.2) [LSB]	vdprintf(GLIBC_2.2) [SUSv4]	vfprintf(GLIBC_2.2) [SUSv4]	vprintf(GLIBC_2.2) [SUSv4]

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

vsnprintf(GLIBC_2.2) [SUSv4]	vsprintf(GLIBC_2.2) [SUSv4]		

An LSB conforming implementation shall provide the architecture specific deprecated functions for Standard I/O specified in Table 10-7, 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 10-7 libc - Standard I/O Deprecated Function Interfaces

tempnam(GLIBC_2.2) [SUSv4]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Standard I/O specified in Table 10-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-8 libc - Standard I/O Data Interfaces

stderr(GLIBC_2.2) [SUSv4]	stdin(GLIBC_2.2) [SUSv4]	stdout(GLIBC_2.2) [SUSv4]	

10.2.5 Signal Handling

10.2.5.1 Interfaces for Signal Handling

An LSB conforming implementation shall provide the architecture specific functions for Signal Handling specified in Table 10-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-9 libc - Signal Handling Function Interfaces

__libc_current_sigrtmax(GLIBC_2.2) [LSB]	__libc_current_sigrtmin(GLIBC_2.2) [LSB]	__sigsetjmp(GLIBC_2.2) [LSB]	__sysv_signal(GLIBC_2.2) [LSB]
__xpg_sigpause(GLIBC_2.2) [LSB]	bsd_signal(GLIBC_2.2) [SUSv3]	psignal(GLIBC_2.2) [SUSv4]	raise(GLIBC_2.2) [SUSv4]
sigaction(GLIBC_2.2) [SUSv4]	sigaddset(GLIBC_2.2) [SUSv4]	sigaltstack(GLIBC_2.2) [SUSv4]	sigandset(GLIBC_2.2) [LSB]
sigdelset(GLIBC_2.2) [SUSv4]	sigemptyset(GLIBC_2.2) [SUSv4]	sigfillset(GLIBC_2.2) [SUSv4]	sighold(GLIBC_2.2) [SUSv4]
sigignore(GLIBC_2.2) [SUSv4]	siginterrupt(GLIBC_2.2) [SUSv4]	sigisemptyset(GLIBC_2.2) [LSB]	sigismember(GLIBC_2.2) [SUSv4]
siglongjmp(GLIBC_2.2) [SUSv4]	signal(GLIBC_2.2) [SUSv4]	sigorset(GLIBC_2.2) [LSB]	sigpause(GLIBC_2.2) [LSB]
sigpending(GLIBC_2.2) [SUSv4]	sigprocmask(GLIBC_2.2) [SUSv4]	sigqueue(GLIBC_2.2) [SUSv4]	sigrelse(GLIBC_2.2) [SUSv4]
sigreturn(GLIBC_2.2) [LSB]	sigset(GLIBC_2.2) [SUSv4]	sigsuspend(GLIBC_2.2) [SUSv4]	sigtimedwait(GLIBC_2.2) [SUSv4]

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

sigwait(GLIBC_2.2) [SUSv4]	sigwaitinfo(GLIBC_2.2) [SUSv4]		

An LSB conforming implementation shall provide the architecture specific deprecated functions for Signal Handling specified in Table 10-10, 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 10-10 libc - Signal Handling Deprecated Function Interfaces

sigpause(GLIBC_2.2) [LSB]			

An LSB conforming implementation shall provide the architecture specific data interfaces for Signal Handling specified in Table 10-11, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-11 libc - Signal Handling Data Interfaces

_sys_siglist(GLIBC_2.3.3) [LSB]			

10.2.6 Localization Functions

10.2.6.1 Interfaces for Localization Functions

An LSB conforming implementation shall provide the architecture specific functions for Localization Functions specified in Table 10-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-12 libc - Localization Functions Function Interfaces

bind_textdomain_codeset(GLIBC_2.2) [LSB]	bindtextdomain(GLIBC_2.2) [LSB]	catclose(GLIBC_2.2) [SUSv4]	catgets(GLIBC_2.2) [SUSv4]
catopen(GLIBC_2.2) [SUSv4]	dcgettext(GLIBC_2.2) [LSB]	dcngettext(GLIBC_2.2) [LSB]	dgettext(GLIBC_2.2) [LSB]
dngettext(GLIBC_2.2) [LSB]	gettext(GLIBC_2.2) [LSB]	iconv(GLIBC_2.2) [SUSv4]	iconv_close(GLIBC_2.2) [SUSv4]
iconv_open(GLIBC_2.2) [SUSv4]	localeconv(GLIBC_2.2) [SUSv4]	ngettext(GLIBC_2.2) [LSB]	nl_langinfo(GLIBC_2.2) [SUSv4]
setlocale(GLIBC_2.2) [SUSv4]	textdomain(GLIBC_2.2) [LSB]		

An LSB conforming implementation shall provide the architecture specific data interfaces for Localization Functions specified in Table 10-13, with the full mandatory functionality as described in the referenced underlying specification.

© 2021 ISO/IEC – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021

Table 10-13 libc - Localization Functions Data Interfaces

_nl_msg_cat_cntr (GLIBC_2.2) [LSB]			

10.2.7 Posix Spawn Option

10.2.7.1 Interfaces for Posix Spawn Option

An LSB conforming implementation shall provide the architecture specific functions for Posix Spawn Option specified in Table 10-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-14 libc - Posix Spawn Option Function Interfaces

posix_spawn(GLIBC_2.15) [SUSv4]	posix_spawn_file_actions_addclose(GLIBC_2.2) [SUSv4]	posix_spawn_file_actions_adddup2(GLIBC_2.2) [SUSv4]	posix_spawn_file_actions_addopen(GLIBC_2.2) [SUSv4]
posix_spawn_file_actions_destroy (GLIBC_2.2) [SUSv4]	posix_spawn_file_actions_init(GLIBC_2.2) [SUSv4]	posix_spawnattr_destroy(GLIBC_2.2) [SUSv4]	posix_spawnattr_getflags(GLIBC_2.2) [SUSv4]
posix_spawnattr_getpgroup(GLIBC_2.2) [SUSv4]	posix_spawnattr_getschedparam(GLIBC_2.2) [SUSv4]	posix_spawnattr_getschedpolicy(GLIBC_2.2) [SUSv4]	posix_spawnattr_getsigdefault(GLIBC_2.2) [SUSv4]
posix_spawnattr_getsigmask(GLIBC_2.2) [SUSv4]	posix_spawnattr_init(GLIBC_2.2) [SUSv4]	posix_spawnattr_setflags(GLIBC_2.2) [SUSv4]	posix_spawnattr_setpgroup(GLIBC_2.2) [SUSv4]
posix_spawnattr_setschedparam(GLIBC_2.2) [SUSv4]	posix_spawnattr_setschedpolicy(GLIBC_2.2) [SUSv4]	posix_spawnattr_setsigdefault(GLIBC_2.2) [SUSv4]	posix_spawnattr_setsigmask(GLIBC_2.2) [SUSv4]
posix_spawnp(GLIBC_2.15) [SUSv4]			

10.2.8 Posix Advisory Option

10.2.8.1 Interfaces for Posix Advisory Option

An LSB conforming implementation shall provide the architecture specific functions for Posix Advisory Option specified in Table 10-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 10-15 libc - Posix Advisory Option Function Interfaces

posix_fadvise(GLIBC_2.2) [SUSv4]	posix_fallocate(GLIBC_2.2) [SUSv4]	posix_madvise(GLIBC_2.2) [SUSv4]	posix_memalign(GLIBC_2.2) [SUSv4]

© 2021 ISO/IEC – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 23360-3-2:2021