INTERNATIONAL STANDARD

ISO/IEC 23360-1-1

First edition
2021-10

Linux Standard Base (LSB) —

Part 1-1:
Common definitions

Reference number
ISO/IEC 23360-1-1:2021(E)

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): Common Definitions 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 23360-1-1 cancels and replaces ISO/IEC 23360-1:2006, 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.

Contents

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.

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.

The LSB specification set is divided into modules, each of which provides fundamental system interfaces, libraries, and runtime environment upon which all conforming applications and libraries using that module depend.

The modules of the Linux Standard Base are:

- LSB Core - core components
- LSB Desktop - desktop related components
- LSB Languages - runtime languages
- LSB Imaging - printing and scanning
- LSB Trial Use - components that are not yet mandatory

Interfaces described in the LSB Core module specification are supplemented by other LSB module specifications. All other modules depend on the presence of LSB Core.

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. Whenever a section of the common part is supplemented by architecture-specific information, the common part includes a reference to the architecture-specific part. Architecture-specific parts of of an LSB module specification may also contain additional information that is not referenced in the common part.

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.

2 Requirements

2.1 Relevant Libraries

The libraries listed in the following tables shall be available on a Linux Standard Base system, with the specified runtime names. The libraries listed in Table 2-2 are architecture specific, but shall be available on all LSB conforming systems under a name specified in each Architecture Specific Part of the LSB Core module.

Table 2-1 LSB Core Module Library Names

Library	Runtime Name
libcrypt	libcrypt.so.1
libdl	libdl.so.2
libgcc_s	libgcc_s.so.1
libncurses	libncurses.so.5
libncursesw	libncursesw.so.5
libnspr4	libnspr4.so
libnss3	libnss3.so
libpam	libpam.so.0
libpthread	libpthread.so.0
librt	librt.so.1
libssl3	libssl3.so
libstdcxx	libstdc++.so.6
libutil	libutil.so.1
libz	libz.so.1

Table 2-2 LSB Core Module Library Names which vary by architecture

Library	Runtime Name
libc	See architecture specific part.
libm	See architecture specific part.
proginterp	See architecture specific part.

Table 2-3 LSB Desktop Module Library Names

Library	Runtime Name
libGL	libGL.so.1
libGLU	libGLU.so.1
libICE	libICE.so.6
libQtCore	libQtCore.so.4
libQtGui	libQtGui.so.4

Library	Runtime Name
libQtNetwork	libQtNetwork.so.4
libQtOpenGL	libQtOpenGL.so.4
libQtSql	libQtSql.so.4
libQtSvg	libQtSvg.so.4
libQtXml	libQtXml.so.4
libSM	libSM.so.6
libX11	libX11.so.6
libXext	libXext.so.6
libXft	libXft.so.2
libXi	libXi.so.6
libXrender	libXrender.so.1
libXt	libXt.so.6
libXtst	libXtst.so.6
libasound	libasound.so.2
libatk-1.0	libatk-1.0.so.0
libcairo	libcairo.so.2
libcairo-gobject	libcairo-gobject.so.2
libcairo-script-interpreter	libcairo-script-interpreter.so.2
libfontconfig	libfontconfig.so.1
libfreetype	libfreetype.so.6
libgdk-x11-2.0	libgdk-x11-2.0.so.0
libgdk_pixbuf-2.0	libgdk_pixbuf-2.0.so.0
libgdk_pixbuf_xlib-2.0	libgdk_pixbuf_xlib-2.0.so.0
libgio-2.0	libgio-2.0.so.0
libglib-2.0	libglib-2.0.so.0
libgmodule-2.0	libgmodule-2.0.so.0
libgobject-2.0	libgobject-2.0.so.0
libgthread-2.0	libgthread-2.0.so.0
libgtk-x11-2.0	libgtk-x11-2.0.so.0
libjpeg	libjpeg.so.62
libpango-1.0	libpango-1.0.so.0
libpangocairo-1.0	libpangocairo-1.0.so.0
libpangoft2-1.0	libpangoft2-1.0.so.0

Library	Runtime Name
libpangoxft-1.0	libpangoxft-1.0.so.0
libpng12	libpng12.so.0
libtiff	libtiff.so.5
libxcb	libxcb.so.1

Table 2-4 LSB Imaging Module Library Names

Library	Runtime Name
libcups	libcups.so.2
libcupsimage	libcupsimage.so.2
libsane	libsane.so.1

Table 2-5 LSB Languages Module Library Names

Library	Runtime Name
libxml2	libxml2.so.2
libxslt	libxslt.so.1

2.2 Relevant Commands

The commands listed in the following tables shall be available on a Linux Standard Base system, with the specified runtime names.

Table 2-6 LSB Core Module Command Names

[	du	install	mv	strings
ar	echo	install_initd	newgrp	strip
at	ed	ipcrm	nice	stty
awk	egrep	ipcs	nl	su
basename	env	join	nohup	sync
batch	expand	kill	od	tail
bc	expr	killall	passwd	tar
cat	false	ln	paste	tee
chfn	fgrep	locale	patch	test
chgrp	file	localedef	pathchk	tic
chmod	find	logger	pax	time
chown	fold	logname	pidof	touch
chsh	fuser	lp	pr	tput
cksum	gencat	lpr	printf	tr
cmp	getconf	ls	ps	true

col	gettext	lsb_release	pwd	tsort
comm	grep	m4	remove_initd	tty
cp	groupadd	mailx	renice	umount
cpio	groupdel	make	rm	uname
crontab	groupmod	man	rmdir	unexpand
csplit	groups	md5sum	sed	uniq
cut	gunzip	mkdir	sendmail	useradd
date	gzip	mkfifo	seq	userdel
dd	head	mknod	sh	usermod
df	hostname	mktemp	shutdown	wc
diff	iconv	more	sleep	xargs
dirname	id	mount	sort	zcat
dmesg	infocmp	msgfmt	split	

Table 2-7 LSB Desktop Module Command Names

fc-cache	fc-match	xdg-desktop-menu	xdg-icon-resource	xdg-open
fc-list	xdg-desktop-icon	xdg-email	xdg-mime	xdg-screensaver

Table 2-8 LSB Imaging Module Command Names

foomatic-rip	gs			

Table 2-9 LSB Languages Module Command Names

perl	python			

2.3 LSB Implementation Conformance

A conforming implementation is necessarily architecture specific, and must provide the interfaces specified by both the generic LSB specifications and the applicable architecture specific part.

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 part of the LSB Core Specification for a given target processor architecture describes a minimum acceptable processor. The implementation shall provide all features of this processor, whether in hardware or through emulation transparent to the application.

- The implementation shall be capable of executing compiled applications having the format and using the system interfaces described in this 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.
- The implementation shall report whether supports for each of the modules constituting this specification is currently available, with the exception of the Trial Use module, which need not be reported. At a minimum, this reporting shall be performed using the **lsb_release** command described in the LSB Core module specification.

Rationale: An implementation must support all modules described as mandatory in this specification. However, excepting the LSB Core module, which is always required, the support for a module may not be installed or enabled. The intent of this clause is to indicate a run-time query mechanism to determine the status of module support.

2.4 LSB Application Conformance

A conforming application containing object files is necessarily architecture specific, and must conform to both the generic LSB Core module 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 module specification (LSB Core - Generic).

In addition, the application may optionally conform to one or more additional LSB module specifications.

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.

Applications distributed using the packaging specification described in the generic LSB Core specification (LSB Core - Generic) may, in addition to other package dependencies described in this specification, declare a dependency on "lsb" with a version of 5.0.

> **Implementation Note:** Application dependencies should generally be as limited as possible. For example, if a 64-bit POWER application only depends on items from the core specification, a dependency on "lsb-core-ppc64" may be more appropriate than a dependency on "lsb". The latter dependency could cause an implementation to install a number of other modules that may not be necessary to execute this application.

3 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/

3.1

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.

3.2

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.

3.3

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.

3.4

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.

3.5

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.

3.6

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.

3.7

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.

4 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.