TECHNICAL SPECIFICATION

ISO/IEC TS 18822

First edition
2015-07-01

Programming languages — C++ — File System Technical Specification

Languages de programmation — C++ — Spécification technique de système de fichiers

Reference number
ISO/IEC TS 18822:2015(E)

© ISO/IEC 2015



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2015

All rights reserved. Unless otherwise specified, 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
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

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

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

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

In other circumstances, particularly when there is an urgent market requirement for such documents, the joint technical committee may decide to publish other types of document:

— an ISO/IEC Publicly Available Specification (ISO/IEC PAS) represents an agreement between technical experts in an ISO/IEC working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;

— an ISO/IEC Technical Specification (ISO/IEC TS) represents an agreement between the members of the joint technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/IEC PAS or ISO/IEC TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/IEC PAS or ISO/IEC TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

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.

ISO/IEC TS 18822 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

© ISO/IEC 2015 – All rights reserved

Contents

Contents
1 Scope
2 Conformance
  2.1 POSIX conformance
  2.2 Operating system dependent behavior conformance
  2.3 File system race behavior
3 Normative references
4 Terms and definitions
  4.1 absolute path
  4.2 canonical path
  4.3 directory
  4.4 file
  4.5 file system
  4.6 file system race
  4.7 filename
  4.8 hard link
  4.9 link
  4.10 native encoding
  4.11 native pathname format
  4.12 NTCTS
  4.13 operating system dependent behavior
  4.14 parent directory
  4.15 path
  4.16 pathname
  4.17 pathname resolution
  4.18 relative path
  4.19 symbolic link
5 Requirements
  5.1 Namespaces and headers
  5.2 Feature test macros
6 Header `<experimental/filesystem>` synopsis
7 Error reporting
8 Class `path`
  8.1 `path` generic pathname format grammar
  8.2 `path` conversions
    8.2.1 `path` argument format conversions
    8.2.2 `path` type and encoding conversions
  8.3 `path` requirements
  8.4 `path` members
    8.4.1 `path` constructors
    8.4.2 `path` assignments

© ISO/IEC 2015 – All rights reserved

8.4.3 `path` appends
8.4.4 `path` concatenation
8.4.5 `path` modifiers
8.4.6 `path` native format observers
8.4.7 `path` generic format observers
8.4.8 `path` compare
8.4.9 `path` decomposition
8.4.10 `path` query
8.5 `path` iterators
8.6 `path` non-member functions
8.6.1 `path` inserter and extractor
8.6.2 `path` factory functions
9 Class `filesystem_error`
9.1 `filesystem_error` members
10 Enumerations
10.1 Enum class `file_type`
10.2 Enum class `copy_options`
10.3 Enum class `perms`
10.4 Enum class `directory_options`
11 Class `file_status`
11.1 `file_status` constructors
11.2 `file_status` observers
11.3 `file_status` modifiers
12 Class `directory_entry`
12.1 `directory_entry` constructors
12.2 `directory_entry` modifiers
12.3 `directory_entry` observers
13 Class `directory_iterator`
13.1 `directory_iterator` members
13.2 `directory_iterator` non-member functions
14 Class `recursive_directory_iterator`
14.1 `recursive_directory_iterator` members
14.2 `recursive_directory_iterator` non-member functions
15 Operational functions
15.1 Absolute
15.2 Canonical
15.3 Copy
15.4 Copy file
15.5 Copy symlink
15.6 Create directories
15.7 Create directory
15.8 Create directory symlink
15.9 Create hard link
15.10 Create symlink

© ISO/IEC 2015 – All rights reserved

15.11 Current path
15.12 Exists
15.13 Equivalent
15.14 File size
15.15 Hard link count
15.16 Is block file
15.17 Is character file
15.18 Is directory
15.19 Is empty
15.20 Is fifo
15.21 Is other
15.22 Is regular file
15.23 Is socket
15.24 Is symlink
15.25 Last write time
15.26 Permissions
15.27 Read symlink
15.28 Remove
15.29 Remove all
15.30 Rename
15.31 Resize file
15.32 Space
15.33 Status
15.34 Status known
15.35 Symlink status
15.36 System complete
15.37 Temporary directory path

© ISO/IEC 2015 – All rights reserved

1 Scope [fs.scope]

1 This Technical Specification specifies requirements for implementations of an interface that computer programs written in the C++ programming language may use to perform operations on file systems and their components, such as paths, regular files, and directories. This Technical Specification is applicable to information technology systems that can access hierarchical file systems, such as those with operating systems that conform to the POSIX (3) interface. This Technical Specification is applicable only to vendors who wish to provide the interface it describes.

2 Conformance [fs.conformance]

1 Conformance is specified in terms of behavior. Ideal behavior is not always implementable, so the conformance sub-clauses take that into account.

2.1 POSIX conformance [fs.conform.9945]

1 Some behavior is specified by reference to POSIX (3). How such behavior is actually implemented is unspecified.

2 [*Note:* This constitutes an "as if" rule allowing implementations to call native operating system or other API's. —*end note*]

3 Implementations are encouraged to provide such behavior as it is defined by POSIX. Implementations shall document any behavior that differs from the behavior defined by POSIX. Implementations that do not support exact POSIX behavior are encouraged to provide behavior as close to POSIX behavior as is reasonable given the limitations of actual operating systems and file systems. If an implementation cannot provide any reasonable behavior, the implementation shall report an error as specified in § 7.

4 [*Note:* This allows users to rely on an exception being thrown or an error code being set when an implementation cannot provide any reasonable behavior. — *end note*]

5 Implementations are not required to provide behavior that is not supported by a particular file system.

6 [*Example:* The FAT file system used by some memory cards, camera memory, and floppy discs does not support hard links, symlinks, and many other features of more capable file systems, so implementations are not required to support those features on the FAT file system. —*end example*]

2.2 Operating system dependent behavior conformance [fs.conform.os]

1 Some behavior is specified as being operating system dependent (4.13). The operating system an implementation is dependent upon is implementation defined.

© ISO/IEC 2015 – All rights reserved

2 It is permissible for an implementation to be dependent upon an operating system emulator rather than the actual underlying operating system.

2.3 File system race behavior [fs.race.behavior]

1 Behavior is undefined if calls to functions provided by this Technical Specification introduce a file system race (4.6).

2 If the possibility of a file system race would make it unreliable for a program to test for a precondition before calling a function described herein, *Requires* is not specified for the function.

3 [*Note:* As a design practice, preconditions are not specified when it is unreasonable for a program to detect them prior to calling the function. —*end note*]

3 Normative references [fs.norm.ref]

1 The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2 • ISO/IEC 14882, *Programming Language C++*

3 • ISO/IEC 9945, *Information Technology — Portable Operating System Interface (POSIX)*

4 [*Note:* The programming language and library described in ISO/IEC 14882 is herein called *the C++ Standard*. References to clauses within the C++ Standard are written as "C++14 §3.2". Section references are relative to N3936.

5 The operating system interface described in ISO/IEC 9945 is herein called *POSIX*. —*end note*]

6 This Technical Specification mentions commercially available operating systems for purposes of exposition. [footnote]

7 Unless otherwise specified, the whole of the C++ Standard's Library introduction (C++14 §17) is included into this Technical Specification by reference.

8 [footnote] POSIX® is a registered trademark of The IEEE. MAC OS® is a registered trademark of Apple Inc. Windows® is a registered trademark of Microsoft Corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of these products.

© ISO/IEC 2015 – All rights reserved

4 Terms and definitions [fs.definitions]

1 For the purposes of this document, the terms and definitions given in the C++ Standard and the following apply.

4.1 absolute path [fs.def.absolute-path]

1 A path that unambiguously identifies the location of a file without reference to an additional starting location. The elements of a path that determine if it is absolute are operating system dependent.

4.2 canonical path [fs.def.canonical-path]

1 An absolute path that has no elements that are symbolic links, and no dot or dot-dot elements (8.1).

4.3 directory [fs.def.directory]

1 A file within a file system that acts as a container of directory entries that contain information about other files, possibly including other directory files.

4.4 file [fs.def.file]

1 An object within a file system that holds user or system data. Files can be written to, or read from, or both. A file has certain attributes, including type. File types include regular files and directories. Other types of files, such as symbolic links, may be supported by the implementation.

4.5 file system [fs.def.filesystem]

1 A collection of files and certain of their attributes.

4.6 file system race [fs.def.race]

1 The condition that occurs when multiple threads, processes, or computers interleave access and modification of the same object within a file system.

4.7 filename [fs.def.filename]

1 The name of a file. Filenames dot and dot-dot have special meaning. The following characteristics of filenames are operating system dependent:

2 • The permitted characters. [*Example*: Some operating systems prohibit the ASCII control characters (0x00-0x1F) in filenames. —*end example*].
3 • The maximum permitted length.
4 • Filenames that are not permitted.

© ISO/IEC 2015 – All rights reserved