

IEEE Standards Interpretations for IEEE Std 1003.1-1990 and IEEE Std 2003.1-1992

Circuits and Devices

Communications Technology

IEEE Computer Society

Sponsored by the
Portable Applications Standards Committee

*Electromagnetics and
Radiation*

Energy and Power

Industrial Applications

*Signals and
Applications*

*Standards
Coordinating
Committees*

IEEE Std 1003.1/2003.1/INT, March 1994 Edition



Published by the Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA.

March 31, 1994

IEEE Standards Interpretations

for IEEE Std 1003.1-1990 and IEEE Std 2003.1-1992

Sponsor

**Portable Applications Standards Committee
of the
IEEE Computer Society**

Abstract: The Portable Applications Standards Committee of the IEEE Computer Society carried out a series of analyses of various problems encountered by users of IEEE Std 1003.1-1990, IEEE Standard for Information Technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) and IEEE Std 2003.1-1992, IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX—Part 1: System Interfaces. The results of its deliberations are presented in this document. The intent is to give the POSIX community reasonable ways of interpreting unclear portions of these standards.

Keywords: computer, computer environments, interfaces, interpretations, portable operating system interface, POSIX, testing, test methods

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1994 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1994. Printed in the United States of America.

ISBN 1-55937-410-1

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards Interpretations

Occasionally, questions may arise regarding the meaning of portions of standards as they relate to specific applications. Such requests for interpretations should ask for clarifications of the exact nature of the contents of the standard.

Questions relating to such interpretations are reviewed and evaluated by a balance of members representing the specific committee interests. This officially formed interpretations subgroup creates and circulates a draft interpretation among its members. It transmits a final interpretation to the party initiating the request only after consensus has been achieved. The interpretation is also given to the standards-developing committee to consider addressing it in a supplement or the next revision to the standard.

Interpretations are issued to explain and clarify the intent of the standard and are not intended to constitute an alteration to the original standard or to supply consulting information. The interpretations subgroup cannot make new rules to fit situations not yet covered in the standard, even if the investigations of the subgroup lead it to conclude that the requirement is incomplete or in error. Changes to the standard are made only through revisions or supplements to the standard.

It is recognized that requests are frequently received that are partially or totally requests for information rather than requests for an interpretation. It is inappropriate to issue an official interpretation to answer such requests. The interpretations subgroup may, however, find from its research that the literal printing of the standards text is not identical to that approved by the standards developers and may issue an editorial correction as a part of its interpretation.

The original interpretations requests in this document have been lightly edited to remove extraneous matter and to focus on the problem presented. Some illustrations have been redrawn for publication. With these exceptions, requests are in the form received.

Procedure for Requesting an Interpretation

Requests for interpretations should be addressed to

Secretary, IEEE Standards Board
IEEE Standards Department
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331

Requests for interpretations should include

- a) The specific designation of the standard, including the year of publication.
- b) The specific subclause being questioned.
- c) The applicable conditions for the case in question.

Line drawings should be black ink or excellent black pencil originals. Photos should be black-and-white glossy prints. These illustrations must be reproduced for committee circulation and eventually will be used to supplement the text of the next edition. Clear diagrams and pictures will make the work of interpretation easier and more valuable to users.

Requests, including all supplementary material, must be in a form that is easily reproducible. If suitable for consideration, requests will be sent to the interpretations subgroup. After consideration by the subcommittee, which may involve many exchanges of correspondence, the inquirer will be notified of the decision of the subgroup. Decisions will be published from time to time in cumulative form and may be ordered from the IEEE.

Introduction

After approval and publication of a standard, questions may arise regarding the meaning of portions of standards as they relate to users of the standard. In the case of IEEE Std 1003.1-1990 and IEEE Std 2003.1-1992, these questions arose from both system implementors, application authors, and test suite developers. When a need for interpretation is brought to the attention of the IEEE, the Institute initiates action to prepare appropriate responses.

IEEE Std 1003.1-1990 (POSIX.1) and IEEE Std 2003.1-1992 were developed by the Portable Applications Standards Committee (PASC) of the IEEE Computer Society. IEEE Std 2003.1-1992 is a test method binding to the content of IEEE Std 1003.1-1990. Each interpretation request is forwarded to the members of the PASC interpretations group and a subgroup is invited to generate a draft interpretation for review. The draft interpretation is made available to the whole interpretations group for review and comment. When consensus has been reached within the interpretations group, the IEEE transmits the final interpretation to the party initiating the request.

It is recognized that requests are frequently received that are partially or totally requests for information rather than requests for interpretation. The IEEE considers it inappropriate to issue an official interpretation to answer such requests. This includes requests for advice as to the means by which particular facilities described in the standard could be implemented.

Every IEEE standard is subjected to review at least every five years for revision or reaffirmation. The revision process takes input from a wide variety of sources in order to develop a revised standard. One of these sources is the interpretations that have been issued against the extant version of the standard. It is not to be expected that the text of the interpretation will appear in the revised standard, but rather that the revision will address the issues raised by the interpretation request through the consensus process. The revision process is allowed to resolve ambiguities in the original standard through textual change; this capability is not allowed within the interpretations process.

This document contains a compilation of all of the interpretations completed against IEEE Std 1003.1-1990 and IEEE Std 2003.1-1992 as of February 1994. The IEEE will provide details of interpretation requests received that have not yet appeared in a publication. Where a formal interpretation has been completed, this will include the text of the interpretation.

At the time that this compilation of interpretations was published, the following members had participated in the IEEE Std 1003.1-1990/IEEE Std 2003.1-1992 interpretations group:

Portable Applications Standards Committee

Chair:	Jim Isaak
Vice-Chair, Interpretations:	Andrew Josey

Mark Brown	Chuck Karish	Tom Robinson
Dawn Burnett	Jeff Kimmel	Seth Rosenthal
Tony Cincotta	Martin Kirk	Carol Schaefer
Lee Damico	David Korn	Keith Stobie
Dave Decot	Shane McCarron	Donn Terry
David Emery	Jim McGuinness	Andrew Twigger
John Farley	J. Jay Meyer	Paul Wanish
Hal Jespersen	Martha Nalebuff	David Willcox
Lowell Johnson	Paul Rabin	Fred Zlotnick
Erik Kaae		John Zolnowsky

Mary Lynne Nielsen
IEEE Standards Project Editor

Interpretations for IEEE Std 1003.1-1990

Interpretation Number: 1
Topic: Behavior of *rename()*
Relevant Clauses: 5.5.3.3
Classification: Editorial defect

Interpretation request

Ambiguity in 5.5.3.3—*rename()*

The words “if either exists” (page 100, line 589) appear to exclude the case where the *old* and *new* file did not exist prior to the function call. Is it not the case that if the function call fails, the implementation is always required to ensure that both the *old* and *new* file states are identical to prior to the call and neither is either created or modified?

A further consideration are implementations that allow *rename()* to be used across file systems by copying rather than linking, and where cleanup and atomicity is critical.

Interpretation for IEEE Std 1003.1-1990

If a call to *rename(old, new)* returns -1, then the implementation shall in all cases ensure that neither *old* nor *new* is created or modified. In particular, if neither *old* nor *new* exists prior to the call to *rename()*, then neither *old* nor *new* shall be created by the call.

Implementations that support *rename()* across file systems are bound by the same semantic requirements for such a call to *rename()* as for a call to *rename()* within a file system.

Rationale for interpretation

As is pointed out in the interpretation request, the standard is quite clear and unambiguous in the case where either *old* or *new* (or both) exist prior to the call. The only case at issue is when neither exists. The language in 5.5.3.3 (which is new in the 1990 revision of the standard) states:

If -1 is returned, neither the file named by *old* nor the file named by *new*, if either exists, shall be changed by this function call.

This does not explicitly state what must occur when neither *old* nor *new* exists.

The interpretation is based on 5.5.3.2, which states (in part):

The *rename()* function changes the name of the file. The *old* argument points to the pathname of the file to be renamed. The *new* argument points to the new pathname of the file.

The *rename()* function is also specified in the C Standard (ANSI X3.159-1989), which in Section 4.9.4.2 states (in part):

The *rename* function causes the file whose name is the string pointed to by *old* to be henceforth known by the name given by the string pointed to by *new*. The file named *old* is no longer accessible by that name.

Thus, *rename()* changes file names, but does not change files. Note that in the descriptions of other functions that resolve pathnames but do not create file system objects, the semantics do not explicitly state that the named file must not be created. Yet to create such a file would be considered a semantic error. Examples include *unlink()*, *stat()*, *chown()*, and *pathconf()*. On the other hand, those interfaces that are

explicitly designed to create file system objects [such as *open()*, *mkdir()*, and *mkfifo()*] document that if -1 is returned, nothing is created.

Given the description of the *rename()* function in IEEE Std 1003.1-1990 and ANSI X3.159-1989, it falls into the same category as *unlink()*, *stat()*, etc. Since file creation is not part of the semantic requirements of *rename()*, there is no need to document the implicit requirement that a call that fails must not create any extraneous files.

Interpretation Number: 3
Topic: EIO missing from tc* functions
Relevant Clauses: 7.1.1.4
Classification: No change required

Interpretation request

Subclause 7.1.1.4 (page 131, lines 88–90) states:

Certain calls that set terminal parameters are treated in the same fashion as `write`, except that TOSTOP is ignored; that is, the effect is identical to that of terminal writes when TOSTOP is set.

In the preceding paragraph, the final sentence says:

If TOSTOP is set, and the process group of the writing process is orphaned, and the writing process is not ignored or blocking SIGTTOU, then `write()` returns -1 with *errno* set to [EIO], and no signal is sent.

Routines that fit this description are

`tcdrain()`, page 145, subclause 7.2.2
`tcflow()`, page 145, subclause 7.2.2
`tcflush()`, page 145, subclause 7.2.2
`tcsendbreak()`, page 145, subclause 7.2.2
`tcsetattr()`, page 143, subclause 7.2.1

Why is [EIO] not included in the error subclause of any of these interfaces? Must they return -1 with *errno* set to [EIO] in the given circumstances? May they?

Interpretation for IEEE Std 1003.1-1990

The requirement in 7.1.1.4, lines 88–90, refers to “calls that set terminal parameters.” The list of functions for which this requirement is valid is limited to `tcsetattr()`. For the case of `tcsetattr()`, the implementation must return [EIO] in the circumstances described as a consequence of lines 88–90 and the description of [EIO] on page 121, lines 280–285.

Rationale for interpretation

Lines 84–87 refer only to the `write()` system call, and not explicitly to the `tcdrain()`, `tcflow()`, `tcflush()`, and `tcsendbreak()` functions. Although they appear to be implementable in terms of `write()`, this is not required by the standard. Were this to be required, language similar to that in 8.2.3 would be included, i.e., “the underlying function is. . .” Therefore, they are not required to return [EIO] by the current standard. They may, however, return [EIO] under the conditions laid out in 2.4, lines 521–526.

Interpretation Number: 5
Topic: Root directory
Relevant Clauses: 2.2.2.74, 3.1.1.2, B.2.3.7
Classification: Editorial defect

Interpretation request

In 2.2.2.74, the definition of root directory is given as

root directory: A directory, associated with a process

In 3.1.1.2, the standard states (page 42, lines 33–36):

All other process characteristics defined by this part of ISO/IEC 9945 shall be the same in the parent and child processes. The inheritance of process characteristics not defined by this part of ISO/IEC 9945 is unspecified by this part of ISO/IEC 9945, but should be documented in the system documentation.

In B.2.3.7, the rationale states (page 211, lines 1131–1132):

The initialization of the root directory of a process is implementation defined.

According to the normative text, however, either the root directory is inherited on *fork()*, or it is unspecified. It is certainly not implementation defined. Since the definition of root directory identifies it as associated with a process, it seems as though inheritance is specified. Is the rationale in error? Does “initialization” refer to something other than “inheritance”?

Interpretation for IEEE Std 1003.1-1990

The rationale is in error.

Rationale for interpretation

The standard requires the root directory to be inherited in the child process after a call to *fork()*. An application cannot make any assumption about its root directory on invocation (nor can a user upon login). In particular, it is not safe to assume that all processes have the same root.

Interpretation Number: 6
Topic: Order of error numbers
Relevant Clauses: 2.4, 3.1.2.4
Classification: No change required

Interpretation request

In 2.4, the standard states (page 23, lines 508–510):

If more than one error occurs in processing a function call, this part of ISO/IEC 9945 does not define in what order the errors are detected; therefore, any one of the possible errors may be returned.

In 3.1.2.4, the standard states (page 45, lines 178–182 and page 46, lines 195–196):

[EACCES] Search permission is denied for a directory listed in the path prefix of the new process image file, or the new process image file denies execution permission, or the new process image file is not a regular file and the implementation does not support execution of files of its type.

[ENOEXEC] The new process image has the appropriate access permissions, but is not in the proper format.

The definition of the error condition corresponding to [ENOEXEC] excludes the possibility that it can co-occur with [EACCES]. Nevertheless, it seems clearly to violate the spirit of the more general requirement, that the order of detection of error conditions not be specified. Is there a conflict here?

Interpretation for IEEE Std 1003.1-1990

If an ENOEXEC error can be generated, then the file must have the correct access permissions, so the EACCES error cannot occur. If the file does not have the correct access permissions, then EACCES will occur. There is no conflict.

Rationale for interpretation

The wording in 2.4 applies only to errors that can co-occur. The wording in 3.1.2.4 indeed excludes the possibility that the [ENOEXEC] and [EACCES] errors can co-occur. This simply means that the wording in 2.4 does not apply to this case. There is no “spirit” of 2.4 that is violated here. There are several places in the standard where one error condition is explicitly worded so that it cannot co-occur with another error condition. This is often necessary either to give the caller useful feedback on error conditions or to prevent violations of access restrictions. The purpose of the wording in 2.4 is to avoid implicit and unnecessary requirements on implementations, and the assumption of such requirements by application writers. The purpose is not to prevent the standard from making explicit, useful requirements.

Interpretation Number: 7
Topic: Ignoring errors
Relevant Clauses: 6.4.1.2, 6.4.2.2, B.6.4
Classification: No change required

Interpretation request

In 6.4.1.2 (page 116, line 116), the standard states:

If *nbyte* is zero, the *read()* function shall return zero and have no other results.

In 6.4.2.2 (page 118, lines 190–191), the standard states:

If *nbyte* is zero and the file is a regular file, the *write()* function shall return zero and have no other results.

In B.6.4 (page 266, lines 3433–3438), the rationale states:

POSIX.1 requires that no action be taken when *nbyte* is zero. This is not intended to take precedence over detection of errors (such as invalid buffer pointers or file descriptors). This is consistent with the rest of POSIX.1, but the phrasing here could be misread to require detection of the zero case before any other errors. A value of zero is considered a correct value, for which the semantics are a no-op.

Rather than being a misreading, the interpretation of 6.4.1.2 and 6.4.2.2 that other errors shall be ignored seems to be the only possible interpretation permitted by the normative text. Is the rationale in error?

Interpretation for IEEE Std 1003.1-1990

The text in 6.4.1.2 and 6.4.2.2 describes the behavior only in the absence of error conditions. The rationale is therefore correct.

Rationale for interpretation

By convention, the description describes behavior only in the absence of error conditions, unless explicitly stated. It is expected that a future issue of the standard will clarify this.

The original intention in this case was to leave error checking optional.