IEEE Standards Interpretations for IEEE Std 2003.1™-1992 IEEE Standard for Information Technology—Test Methods for Measuring Conformance to POSIX®--Part1: System Interfaces

These are interpretations of IEEE Std 2003.1-1992.

9 August 1996

Interpretation Request #1
Topic: fflush & EBADF **Relevant Clauses:** 8.1.11.1
Assertion 6 of subclause 8.1.11.1 of IEEE Std 2003.1-1992 states:
06(A) When the stream pointer argument addresses a file descriptor that is closed, then a call to fflush() returns a value of EOF and sets errno to [EBADF]. Is this correct?

The requirements for error reporting for fflush() are in 8.2.3.11 of 1003.1- 1990:
If any of the functions above return an error indication, the value of errno shall be set to indicate the error condition. If that error condition is one that this part of ISO/IEC 9945 specifies to be detected by one of the corresponding underlying functions, the value of errno shall be the same as the value specified for the underlying function.

The C Standard says only this about error returns from fflush():
The fflush function returns EOF if a write error occurs, otherwise zero.

The C Standard does not specify when a write error occurs. The unconditional requirement that fflush() detect a bad file descriptor would seem to go beyond the requirements of POSIX.1. For example, if no data is present in the buffer for the stream on which fflush() is called, it is conforming for fflush() to return success without making any attempt to access the file descriptor associated with the stream. Since there is no need to call write() (the underlying function for fflush()) there is no guarantee that there will be a write error.

My feeling is that an assertion that referred to data in the buffer would be conforming:
06(C) If the implementation supports buffered streams:

- When the stream pointer argument addresses a file descriptor that is closed and there is data in the buffer for the stream pointed to, then a call to fflush() returns a value of EOF and sets errno to [EBADF].

The IEEE Std 2003.1-1992 has the following assertion for fflush():
06(A) When a stream pointer argument addresses a file descriptor that is closed, then a call to fflush() returns a value of EOF and sets errno to [EBADF]. The corresponding test in the NIST PCTS 151-2 beta test suite has a test strategy as follows: fd = Zopen(path, O_RDONLY | O_CREAT, PROT_ALL); sfd = Zfdopen(fd, "r"); Zclose(fd); expecting(EOF); expecting(EBADF); Zfflush(sfd);

The problem is the requirement that this strategy places that fflush() return -1 in this case when we have a read stream.

For write streams, fflush() has to be sure that any buffered data is written. For read streams, fflush() has to discard any buffer contents and adjust the current seek address. For read/write streams, fflush() has to enable a change in I/O direction. Given that in the above code fragment there is no underlying operation to perform (there isn't even a buffer, let alone any contents to ignore), there should not be a requirement to return an EBADF error. Only the C standard and 1003.1 own the rules for fflush(). There is no requirement that we can find in either that corresponds to the above. Moreover, the above would require adding an otherwise useless system call to fflush(). Think about the precedent that this would set: Theoretically, this would imply that, unless granted immunity, each stdio function would have to do test-the-file- descriptor code whenever it didn't happen to get to an underlying operation that would do the verification already. All the test assertions in 2003.1 are supposed to follow directly from 1003.1 statements. The assertion quoted above has no basis in POSIX.1. We recommend that this assertion not apply to read streams.

The interpretation for IEEE PASC 1003.1-90 #23 applies to these requests: PASC 2003.1-92 #1, and PASC 2003.1-92 #14.

Rationale for Interpretation: See PASC 1003.1-90 #23. See interpretation for 2003.1-92#14.

Interpretation Response #1
Topic: fflush & EBADF **Relevant Sections:** 8.1.11.1
The IEEE Std 2003.1-1992 has the following assertion for fflush():
06(A) When a stream pointer argument addresses a file descriptor that is closed, then a call to fflush() returns a value of EOF and sets errno to [EBADF]. The corresponding test in the NIST PCTS 151-2 beta test suite has a test strategy as follows: fd = Zopen(path, O_RDONLY | O_CREAT, PROT_ALL); sfd = Zfdopen(fd, "r"); Zclose(fd); expecting(EOF);

expecting(EBADF); Zfflush(sfd);

The problem is the requirement that this strategy places that fflush() return -1 in this case when we have a read stream.

Rationale: For write streams, fflush() has to be sure that any buffered data is written. For read streams, fflush() has to discard any buffer contents and adjust the current seek address. For read/write streams, fflush() has to enable a change in I/O direction. Given that in the above code fragment there is no underlying operation to perform (there isn't even a buffer, let alone any contents to ignore), there should not be a requirement to return an EBADF error. Only the C standard and 1003.1-1990 own the rules for fflush().

There is no requirement that we can find in either that corresponds to the above. Moreover, the above would require adding an otherwise useless system call to fflush(). Think about the precedent that this would set: Theoretically, this would imply that, unless granted immunity, each stdio function would have to do test-the-file- descriptor code whenever it didn't happen to get to an underlying operation that would do the verification already.

All the test assertions in 2003.1-1992 are supposed to follow directly from 1003.1 statements. The assertion quoted above has no basis in POSIX.1. We recommend that this assertion not apply to read streams.

The interpretation for IEEE PASC 1003.1-90 #23 applies to these requests: 2003.1-1992 #1, and 2003.1992 #14.

Rationale for Interpretation
See IEEE 1003.1-1990 #23. IEEE 2003.1 #1 and #14 (are actually one and the same) PASC Interpretation reference 2003.1-1992 #1,#14 Classification: duplicate request These are duplicates of PASC interpretation PASC 1003.1-90 #23. Refer to PASC 1003.1-90 #23 for the response, published in 1Q94. See also interpretation for 2003.1-92 #1.