IEEE Standards Interpretation for IEEE Std 1003.1™-1990 IEEE Standard for Information Technology--Portable Operating System Interfaces (POSIX®)

Interpretation Request #58
Topic: fseek and ESPIPE **Relevant Sections:** 8.1

When the underlying open file descriptor references a pipe or FIFO, then a call to fseek() sets errno to ESPIPE, returns a non-zero value and the value of the file pointer is unchanged.

This particular issue has uncovered many ambiguities within POSIX and within many implementations. The following is a list of questions and some insights that may have a bearing on the issue.

(1) Is a pipe or FIFO a device or a special file type?
There seems to be some sloppy wording in POSIX regarding "devices which are incapable of seeking" and whether these include pipes and FIFOs. The definition of a device is "A computer peripheral or an object that appears to the application as such" and that of a FIFO is "A FIFO special file is a type of file". We understand from these definitions that a pipe or a FIFO cannot be classified as a device. If this is incorrect, then the behaviour of fseek() on a pipe or FIFO is implementation defined and just about any behaviour is acceptable after an attempt to perform a file positioning operation on a pipe or FIFO. We understand that the behaviour of a file positioning operation on a pipe or FIFO is well defined to indicate an ESPIPE error.

(2) How does a file positioning operation on a pipe or FIFO affect subsequent read operations?
POSIX states in the definition of file offset "There is no file offset specified for a pipe or FIFO". In tests for lseek() this has been understood by the various test suite developers to mean that subsequent reads from the pipe or FIFO are unaffected by the attempted lseek(), and this test has not given any problems. POSIX.3.1 Draft 13 has

extrapolated this text for lseek() into the equivalent assertion for fseek(). This extrapolation has a further problem in that a stream does not have file offset but has a file-position indicator. So does the POSIX.3.1 assertion really refer to the file position indicator or to the file pointer. The X/Open POSIX.1-90 test suite understands this statement to be the equivalent of the statement for lseek(), in that subsequent reads from the stream are unaffected by the attempted fseek(). This test, however, is exhibiting problems and is the subject for this interpretation.

(3) Could the assertion be refering to the underlying file descriptor?
If this were the case, then it would be necessary to access the pipe and FIFO both from a stream (toundertake the fseek()) and from a file descriptor (to undertake the read()) to ensure that data loss did not occur at the file descriptor level. Unfortunately, the synchronisation rules for file handles does not provide a mechanism for handing off file handles for buffered pipes. This makes it impossible to produce a POSIX conforming application to test this understanding of the assertion, though this doesn't seem to deter all test suite authors! The fact that such an assertion results in a pass is not entirely a surprise and only serves to prove how well behaved implementations are when the behavior is undefined.

(4) Could the assertion be refering to the continued ability to read data from the stream without anyside-effects?
The X/Open POSIX.1-90 test suite believes this to be the case. While it is accepted that the POSIX does not explicitly state that this is the case, it seems valid to argue that side-effects that exhibit data loss after an error condition has been raised should be documented. This data loss seems particularly dangerous since there is no means of recovery once an application has erroneously undertaken an fseek() ona stream. However, these arguments have to be balanced against the fact that there is no explicit mention in the POSIX and in such cases the implementation could be argued to have a degree of freedom. The X/Open proposed resolution is: An fseek() on a pipe or FIFO should return ESPIPE and results in undefined behaviour to the stdio stream.

Interpretation Response
An fseek() on a pipe or FIFO need not return an error. If the fseek() detects the error, then it must fail with errno set to ESPIPE. The POSIX.1 standard does not specify the state of the stream after such an error occurs.

The language in POSIX.1 does not support the cited assertion from 2003.1. The language in POSIX.1 could be clearer, and this concern over clarity has been forwarded to the sponsors.

Rationale for Interpretation
None.