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

Interpretation Request #16
Topic: EISDIR **Relevant Sections:** 5.3.1.4 **Classification:** No change.

Section 5.3.1.4 provides the errno EISDIR for the open() function when the file being opened enabling writes and the file is a directory. Section 2.4 (lines 521-524) allows implementations to generate error numbers listed, under circumstances other than those described "if and only if..."

We would like to know whether or not a conforming implementation can return an EISDIR error for opening a directory for reading. It does seem apparent that a portable application cannot use open, read, and close on a directory because the standard does not define the format of a directory file, and without that information, one cannot interpret the results of the read.

If an implementation chooses to use file descriptors to implement directory functions and a directory file descriptor number is passed to close() or dup() or either parameter of dup2(), are these functions permitted to fail? Our implementation of opendir(), readdir(), rewinddir() and closedir() calls for these functions to be kernel interfaces, so we could be fairly intelligent in preventing a program from doing something that does not make sense(i.e. reading a file whose contents cannot be interpretated). We'd like to know what the standard permits.

Interpretation Response
An implementation must permit an application to open for reading a filename referring to a directory. An implementation which returns an error indication of EISDIR in this situation is nonconforming as this error is limited to calling open() for write or read/write access. The file descriptor returned by open() can be used normally with other functions that take file descriptors as arguments.

An application which opens a directory with open() and reads it with read() is not necessarily nonconforming, but the resulting contents of the buffer are unspecified by POSIX.1. The standard does not specify a relationship between file descriptors underlying the DIR datatype from opendir() and a file descriptor obtained by calling open().

Rationale for Interpretation

There are other ways a program might use a file descriptor referring to a directory that do not involve read(), such as using fstat() to periodically check whether the modification time of a directory has changed and using this information to trigger rescanning the directory with rewinddir() and readdir().