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

Interpretation Request #66
Topic: inherited file descriptors **Relevant Sections:** not specified

There appears to be a hole in the standard for inherited file descriptors. Specifically, the 1003.1a standard appears not to cover the following: fd=open("log",1); fork(); Both parent and child write N times to fd via write(fd,buf,1); parent and child terminate normally.

No seeks are done. Two aspects to this situation require clarification.

1. First, the fact that there are dependencies between I/O's done by the two processes needs to be documented. For example, an lseek() done by the parent affects the child.

2. In the absence of error conditions in above example, should ***exactly*** 2*N bytes be written to "log"? At issue here is the atomicity of write(). Would a conforming implementation be allowed to do the following on occasion: parent write data to byte K child write data to byte K parent updates file position to K+1 child updates file position to K+1 and hence would write ***less than*** 2*N bytes to "log"? Or, is the combination of writing and updating the file position atomic?

Interpretation Response
(1)This is documented. See fork() page 41 lines 19-21; also see open file description on page 16 lines 242-247. (2) The standard does not speak to this issue, and as such no conformance distinction can be made between alternate implementations based on this. This is being referred to the sponsor.

Rationale for Interpretation
None.

Editorial Note (not part of the interpretation)
The proposed revision IEEE Draft 1003.1a has text to address point 2 - atomicity of writes.