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

Interpretation Request #48

Topic: last close() on terminal **Relevant Sections:** 7.1.1.11 **Classification:** No change

POSIX.1-1990 section 7.1.1.11 Closing a Terminal Device file states:

The last process to close a terminal device file shall cause any output to be sent to the device and any input to be discarded.

Is it required behaviour that in the case that output has previously been suspended by a call to tcflow(), that the close() will allow output to be restarted or is it permissible behaviour for the data to be discarded? X/Open proposes that close() will allow output to be restarted.

Interpretation Response

The standard does not specify that a close() on a terminal device file include the equivalent of a call to tcflow(fd,TCOON). The language of 7.1.1.11 allows, but does not require such an action.

An implementation that discards output at the time the close() is called, after reporting in the return value to the write() call that the data was written does not conform to POSIX.1.

Rationale for Interpretation

Section 7.1.1.8 clearly allows for some buffering to occur on terminal output, but the standard leaves unspecified the detailed behavior of this buffering and its interaction with program-directed flow control (tcflow()) and externally generated flow control. It is worth reiterating that an application has functions such as tcdrain(), tcflush(), and tcflow() available to obtain the detailed behavior it requires.

At the time of last close() on a terminal device, an application relinquishes any ability to exert flow control via tcflow(). Contrary to B.7.1.1.11, the implementation is never permitted to "cause a flush of pending output", if "flush" is taken to mean "discard". In the situation described, the two options are "resume output and wait for it to drain" and "block (until interrupted by a signal)". External flow control could cause the first option to degenerate into the second. One overall intent is that a naive program can have its output directed to a terminal device without danger of truncation from close() being called immediately after successful return from the last write().