

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

Copyright © 2001 by the Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue New York, New York 10016-5997 USA All Rights Reserved.

Interpretations are issued to explain and clarify the intent of a standard and do not constitute an alteration to the original standard. In addition, interpretations are not intended to supply consulting information. Permission is hereby granted to download and print one copy of this document. Individuals seeking permission to reproduce and/or distribute this document in its entirety or portions of this document must contact the IEEE Standards Department for the appropriate license. Use of the information contained in this document is at your own risk.

IEEE Standards Department Copyrights and Permissions 445 Hoes Lane, Piscataway, New Jersey 08855-1331, USA

Interpretation Request #61

Topic: signals and preemption **Relevant Sections:** 3.1.1.2

In the following C program, given that the fork function call completes successfully, and neither the child or parent processes are delivered any signals, does POSIX.1 require that the parent reach the point where it returns 0 from main? Must a signal be deliverable once the child has entered the endless loop? I don't know if I've successfully embodied my questions in code, but basically my questions are: 1) Does POSIX.1 REQUIRE support for pre-emption of one process by another? 2) Are signals required to be deliverable independent of the execution state(s) of the processes that are executing? example C program: `#define A_LONG_TIME 100000 #include <unistd.h> main() { switch (fork()) { case -1: return 1; case 0: /* child is running */ /* child enters endless loop*/ for (;;) ; default: /* parent is running*/ if (sleep(A_LONG_TIME)) return 1; return 0; } }`

Interpretation Response

The standard does not require support for pre-emption of one process by another. The standard is silent on the issue of process scheduling altogether; the only mention the standard makes regarding multiprocess execution is "...both the parent and child processes shall be capable of executing independently before either terminates" (Sec. 3.1.1.2, lines 37-38). This means that it is allowable that in the example, if the child never blocks the parent will never execute, as long as it is capable of executing independently should the child ever block.

Signals are not required to be deliverable independent of the execution state(s) of the processes that are executing. Section 3.3.1.2 discuss 'generation' and 'delivery' as separate events, with the delivery interval (the 'pending' state) being 'undetectable' (and hence to some degree untestable) by the application. The standard is silent on the 'pending' state of the signal; it is unspecified when the pending state is ended for a pro-

cess that is not blocking that particular signal. Hence, there is no requirement for forcing delivery. The exceptions to this are specified in the description of the `kill()` interface, in which delivery is mandatory in the case of a process sending a signal to itself using this interface, and also `sigprocmask()` which has wording that requires signals to be delivered.

Rationale for Interpretation

It was the clear intent of the committee that two processes which pass information back and forth across a pipe would both be able to progress properly. A historical example of this situation is the implementation of the desk calculator `bc`, which converted users infix statements to the Reverse Polish Notation accepted by another desk calculator program `dc`, sent the expression to `dc` via a pipe, and expected the result to be returned via another pipe from `dc`. An implementation which could not support this implementation of `bc` would be non-conforming.

Editorial note (not part of this interpretation)

Consider adding in the next edition of .1 an explicit sample program that passes information back and forth thru a pipe, such that the clear intent is expressed in that program. A conforming application would have to execute that program (as well as meet the current requirements).