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

Interpretation Request #132
Topic: SIGCHLD **Relevant Sections:** 1003.1:1996 p59 lines 161-171 PASC

This interpretation request is being filed in order to bring this discussion into scope for the Austin Group Revision. The following description is extracted from several emails that have discussed this problem (with thanks to Bruce Korb, David Korn, et al).

Question: "Is an implementation that sets the signal handler for SIGCHLD to SIG_DFL in any of the exec family of functions conforming?"
From the email discussion: Bruce Korb: A question occurred to me. If someone does not know that signal handling defaulting to Ignore is different from setting the handling to ignore, then one can inadvertently exec a shell that has SIGCHLD handling set to ignore. I did that myself. If that happens, then how is the shell going to know exit status? If it does not know subprocess exit status the shell won't do anything right. On the other hand, if shells are required to set SIGCHLD handling to SIG_DFL for themselves, then we have a lot of broken shells out there. Not to mention, broken other programs that get surprised when their wait4() calls fail. (It cannot be the fault of the parent process because they cannot always know which calls will spawn a child process. Even if they did, you'll wind up with pug-ugly code and timing hassles trying to dance around calls that may or may not spawn children.)

David Korn: System V Release 2 had a signal named SIGCLD and BSD 4.1 has a signal named SIGCHLD which were both referred to as sig child. Both of these were added to earlier version of UNIX so both were added in a way that would not affect existing programs by default. Thus, their default behavior was to behave is if the signal were ignored.

Thus, no blocking call would be generated by either of these signals. The SIGCHLD or

SIGCLD were used for implementing job control. Since I had implemented job control for both BSD and System V, I pointed out to the standards group that except for SIG_DFL, these signals had different semantics. If a signal handler was set for SIGCLD, then a signal would be generated if there were any unreaped child processes. When the signal handler was caught in System V, it was reset by default to SIG_DFL. However, if a process did not reap a child and instead reestablished the signal handler, it would go into an infinite loop since the signal would be generated again. The SIGCLD SIG_IGN behavior was that the system reaped the child when it completed so that the application didn't have to deal with it.

However, I believe that a process blocked in wait() would be awakened, but I am not certain of this. The SIGCHLD signal on the other hand was generated when a child completed if a signal handler was set at that time. No signal would be generated if a signal handler was established while there was waiting children. The SIGCHLD signal was also generated when a child process stopped. I believe that BSD treated SIGCHLD SIG_IGN the same way that it treated SIGCHLD SIG_DFL. The standard adopted the BSD SIGCHLD signal semantics with the following changes:

1. The SA_NOCLDSTOP flag was added so that programs that did not expect a signal on stop would not be affected.
2. The behavior of SIGCHLD was made unspecified when SIG_IGN is specified.

The problem that is being presented is the case in which a process has SIGCHLD set to SIG_IGN and then execs a new process. A conforming application would not set SIGCHLD to SIG_IGN since the standard leaves this behavior unspecified. An application that does set SIGCHLD to SIG_IGN should set it back to SIG_DFL before the call to exec. The standard clearly states that signals set to SIG_IGN by the calling process image shall be set to be ignored by the new process image. However, the fact that the behavior is unspecified, allows an implementation to treat this is if SIG_DFL were set and not automatically reap children, even if setting to SIG_IGN by the process itself would reap children.

Change the sentence "Signals set to be ignored (SIG_IGN) by the calling process image shall be set to be ignored by the new process image." to, "Except for SIGCHLD, signals set to be ignored (SIG_IGN) by the calling process image shall be set to be ignored by the new process image. If SIGCHLD is set to SIG_IGN it shall be set to SIG_DFL in the new process image."

Interpretation Response

The question at hand is whether or not an exec() implementation conforms to the standard if it resets SIGCHLD dispositions from SIG_IGN to SIG_DFL. Two application groupings have been considered in the formulation of this proposed interpretation: - Applications that assume they always inherit SIG_DFL; and - Applications that fork() child processes but never wait() for them, potentially filling up the process table. The standard clearly states that implementation behaviour when SIGCHLD is set to SIG_IGN is unspecified. Yet, the standard also just as clearly states that signals set to be ignored

by the calling process image are to remain set to be ignored by the new process image. This "contradiction" is only apparent, but should be clarified. If this is interpreted to mean that SIGCHLD is required to be ignored by the new process image, then applications in the first grouping above cannot conform to the standard without change. However, the implementation retains an ability to control applications in the second grouping. If the implementation resets SIGCHLD to SIG_DFL, applications in the first grouping are allowed to conform to the standard, but the implementation loses its ability to control applications in the second grouping through the SIGCHLD disposition they inherit. This proposed interpretation takes a somewhat "middle-of-the-road" approach. Whether or not an exec() implementation resets SIGCHLD from SIG_IGN to SIG_DFL would be left as unspecified, including the existence or even nature of any mechanism the implementation would use to decide.

Rationale for Interpretation
This proposed interpretation does not invalidate the conformance of existing implementations or applications. It extends the implementation's ability to deal with the issues involved as it sees fit, by not constraining the implementation to a specific behaviour. The implementation may wish to - unconditionally leave SIGCHLD set to SIG_IGN; or - unconditionally reset SIGCHLD to SIG_DFL; or - provide some mechanism, not specified in this standard, to control inherited SIGCHLD dispositions. This proposed interpretation also negates the need to "invent" some other signal disposition for SIGCHLD.

Notes to Project Editor (not part of the interpretation)
XSH, exec(), DESCRIPTION, page 777 - Line 9894, replace "Signals" with "Except for SIGCHLD, signals" - Line 9896, insert just before "After" at the tail end "If the SIGCHLD signal is set to be ignored by the calling process image, it is unspecified whether the SIGCHLD signal is set to be ignored or to the default action in the new process image."
XSH, posix_spawn(), DESCRIPTION, page 1373 - Line 28599, replace "Signals set to be ignored" with "Except for SIGCHLD, signals set to be ignored" - Add new paragraph after line 28602, reading "If the SIGCHLD signal is set to be ignored by the calling process, it is unspecified whether the SIGCHLD signal is set to be ignored or to the default action in the child process, unless otherwise specified by the POSIX_SPAWN_SETSIGDEF flag being set in the spawn_flags attribute of the object referenced by attrp and the SIGCHLD signal being indicated in the spawn-sigdefault attribute of the object referenced by attrp."

XRAT, B.2.4.3 Signal Actions, page 3389 Insert new paragraph between lines 3638 and 3639, reading "Whether or not an implementation allows SIG_IGN as a SIGCHLD disposition to be inherited across a call to one of the exec() family of functions or posix_spawn() is explicitly left as unspecified. This change was made as a result of IEEE PASC Interpretation #132, and permits the implementation to decide between the following alternatives.

- Unconditionally leave SIGCHLD set to SIG_IGN, in which case the implementation would not allow applications that assume inheritance of SIG_DFL to conform to the stan-

dard without change. The implementation would however retain an ability to control applications that create child processes but never call one of the wait() family of functions, potentially filling up the process table.
- Unconditionally reset SIGCHLD to SIG_DFL, in which case the implementation would allow applications that assume inheritence of SIG_DFL to conform. The implementation would however lose an ability to control applications that spawn child processes but never reap them.
- Provide some mechanism, not specified in this standard, to control inherited SIGCHLD dispositions."

Forwarded to Interpretations Group: 12 Mar 2001 Recirculation posted: 28 Mar 2001 Finalized:10 April 2001