IEEE P1003.2 Draft 11.2 − September 1991

B.3 C Binding for Shell Command Interface

B.3.0.1 C Binding for Shell Command Interface Rationale. *(This subclause is not a part of P1003.2)*

The *system* () and *popen* () functions should not be used by programs that have set user (or group) ID privileges, as defined in POSIX.1 {8}. The *fork* () and *exec* family of functions [except *execlp* () and *execvp* ()], also defined in POSIX.1 {8}, should be used instead. This prevents any unforeseen manipulation of the user's environment that could cause execution of commands not anticipated by the calling program.

If the original and '*popen* ()ed'' processes both intend to read or write or read and write a common file, and either will be using FILE-type C functions [*fread* (), *fwrite* (), etc.], the rules in POSIX.1 {8}8.2.3 must be observed.

B.3.1 C Binding for Execute Command

Function: *system* ()

B.3.1.1 Synopsis

```
#include <stdlib.h>

int system(const char *command );
```

B.3.1.2 Description

This standard requires the *system* () function as described in the C Standard {7}.

The *system* () function shall execute the command specified by the string pointed to by *command* . The environment of the executed command shall be as if a child process were created using the POSIX.1 {8} *fork* () function, and the child process invoked the `sh` utility (see 4.56) using the POSIX.1 {8} *execl* () function as follows:

```
execl(<shell path>, "sh", "-c", command , (char *)0);
```

where *<shell path>* is an unspecified pathname for the `sh` utility.

The *system* () function shall ignore the SIGINT and SIGQUIT signals, and block the SIGCHLD signal, while waiting for the command to terminate. If this might cause the application to miss a signal that would have killed it, then the application should examine the return value from *system* () and take whatever action is appropriate to the application if the command terminated due to receipt of a signal.

The *system* () function shall not affect the termination status of any child of the calling processes other than the process(es) it itself creates.

The *system* () function shall not return until the child process has terminated.