IEEE P1003.2 Draft 11.2 – September 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

B.2.3 Execution-Time Symbolic Constants for Portability Specifications

The macros in Table B-5 can be used by the application at execution time to determine which optional facilities are present. If a macro is defined to have the value −1 in the header <unistd.h>, the implementation shall never provide that feature when the application runs under that implementation. If a macro is defined to have a value other than −1, the implementation shall always provide that feature. If the macro is undefined, then the *sysconf* () function (see B.10.2) can be used to determine if the feature is provided for a particular invocation of the application.

Table B-5 − C Execution-Time Symbolic Constants

Macro Name	Description
_POSIX 2_C_DEV	The system supports the C Language Development Utilities Option (see Annex A)
_POSIX 2_FORT _DEV	The system supports the FORTRAN Development Utilities Option (see Annex C)
_POSIX 2_FORT _RUN	The system supports the FORTRAN Runtime Utilities Option (see Annex C)
_POSIX 2_LOCALEDEF	The system supports the creation of locales as described in 4.35.
_POSIX 2_SW_DEV	The system supports the Software Development Utilities Option (see Section 6)

B.2.4 POSIX. 1 C Numerical Limits

The macros specified in POSIX .1 {8} to provide compile-time values for the configurable variables in Table 7-1 (see 7.8.2) shall also be visible in a POSIX. 2 system. Other macros required by POSIX .1 {8} 2.9 (Numerical Limits) and 2.10 (Symbolic Constants) may also be visible in a POSIX. 2 system.

B.2.4.1 POSIX. 1 C Numerical Limits Rationale. *(This subclause is not a part of P1003.2)*

Subclause 7.8.2 requires that certain POSIX .1 {8} configurable variables be visible in POSIX. 2. Subclause B.2.4 ensures that POSIX. 2 C applications can obtain these variables using the same macros as POSIX .1 {8} C applications. It also allows an implementation to make all of the POSIX .1 {8} macros available even if _POSIX _SOURCE is not set. It also allows an implementation to make all of the POSIX .1 {8} symbols available even if it does not support all of POSIX .1 {8}.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.