IEEE P1003.2b Draft 2 − December 1991

require the symbolic link or contiguous file extensions, but does define a standard way of archiving these files so that all conforming systems can interpret these file types in a meaningful and consistent manner. On a system that does not support extended file types, the format-interpreting utility should do the best it can with the file and go on to the next.

4.51 pwd — Return working directory name

⇒ **4.51.2 pwd Description.** *Replace the entire Description sublcause with:*

> The `pwd` utility shall write to standard output an absolute pathname of the current working directory with all symbolic link components resolved.

Editor's Note: The following rationale will be added to E.4.51, but is kept here with `pwd` *for this draft:*

pwd Rationale. *(This subclause is not a part of P1003.2b)*

HLJ : My version of the KornShell doesn't do this. Is that correct?

4.53 `rm` — Remove directory entries

⇒ **4.53.2 `rm` Description.** *Replace item (2c) with:*

For each entry contained in *file*, other than dot or dot-dot, the four steps listed here [(1)-(4)] shall be taken with the entry as if it were a *file* operand. The `rm` utility shall not traverse directories by following symbolic links into other parts of the hierarchy, but shall remove the links themselves.

Editor's Note: The following rationale will be added to E.4.53, but is kept here with `rm` *for this draft:*

`rm` Rationale. *(This subclause is not a part of P1003.2b)*

The `rm` utility removes symbolic links themselves, not the files they refer to, as a consequence of the dependence on the POSIX.1 {8} *unlink*() functionality, per the Description. When removing hierachies with `−r` or `−R`, the prohibition on following symbolic links has to be made explicit.

4.62 `test` — Evaluate expression

⇒ **4.62.3 `test` Operands.** *Add the following primary in the proper sorted order:*

`–h` *file* True if *file* exists and is a symbolic link.

⇒ **4.62.3 `test` Operands.** *Add the following at the end of the primaries list (before the paragraph that begins* ‘‘A primary can be preceded by . . . ’’

With the exception of the `–h` *file* primary, if a *file* argument is a symbolic link, `test` shall evaluate the expression by resolving the symbolic link and using the file referenced by the link.

Section 5: Revisions to User Portability Utilities Option

5.9 du — Estimate file usage

⇒ **5.9.2 du Description.** *Add a new sentence in the first paragraph, following the sentence beginning with* ‘The du utility, by default ... ’’

When a symbolic link is encountered in the file hierarchy, du shall count and write the name and size of the symbolic link itself (rather than the file referenced by the link) and shall continue with the other files in the directory containing the link, not attempting to follow the link to another portion of the file hierarchy.

5.39 `file` — Determine file type

⇒ **5.14.1 `file` Synopsis.** *Modify the Synopsis to be:*

`file [−h]` [*file* ...]

⇒ **5.14.2 `file` Description.** *Add a new paragraph at the end of the subclause:*

If *file* is a symbolic link, by default the link shall be resolved and `file` shall test the type of file referenced by the symbolic link.

⇒ **5.14.3 `file` Options.** *Replace the entire Options subclause with:*

`−h` When a symbolic link is encountered, the link shall not be resolved and `file` shall identify the file as a symbolic link. If `−h` is not specified and *file* is a symbolic link that refers to a nonexistent file, `file` shall identify the file as a symbolic link, as if `−h` had been specified.

⇒ **5.14.6.1 `file` Standard Output.** *Insert a new paragraph between the second and third (the one beginning '*If the file named ...*'')paragraphs:*

If *file* is identified as a symbolic link (see `−h`), the following alternative output format shall be used:

`"%s: %s %s\n"`, <*file*>, <*type*>, <*contents of link*>

⇒ **5.14.6.1 `file` Standard Output.** *Add the following entry to the table named* `file` *Output Strings:*

If *file* is a	**<*type*> shall contain the string**
symbolic link	`symbolic link to`

Section 6: Revisions to Software Development Utilities Option

There are no revisions to Section 6 (yet).

Annex A
(normative)
Revisions to C Language Development Utilities Option

There are no revisions to Annex A (yet).

Annex B
(normative)

Revisions to C Language Bindings Option

Editor's Note: This annex is new for Draft 3. It is not further diff-marked. 3

B.2.5 Pathname Variable Values

⇒ **B.2.3 Pathname Variable Values.** *Add a new subclause, B.2.5, Pathname Variable Values, as follows:*

The macros in Table B-100 can be used by the application at execution time to determine which optional facilities relate to specific pathnames. A definition of one of the values from Table B-100 shall be omitted from `<limits.h>` on specific implementations where the corresponding value is equal to or greater than the stated minimum, but where the value can vary depending on the file to which it is applied. The actual value supported for a specific pathname shall be provided by the *pathconf* () function.

Table B-100 – C Pathname Variable Values

Name	Description
_POSIX2_SYMLINKS	When referring to a directory. the system supports the creation of symbolic links within that directory; for nondirectory files, the meaning of {POSIX2_SYMLINKS} is undefined.

⇒ **B.10.2 C Binding for Get Numeric-Valued Configurable Variables** *Add the following sentence to the end of the only paragraph in the subclause:*

In addition, the *pathconf* () and *fpathconf* () functions shall support the *name* values in Table B-101, defined in `<unistd.h>`, to provide values for variables in 2.13.3.

Table B-101 – C Bindings for Pathname Configurable Variables

Symbolic Limit	*name* **Value**
{POSIX2_SYMLINKS}	_SC_2_SYMLINKS

Alphabetic Topical Index

E

F

G

H

I

K

L

M

N

O

P

V

W