--

--

IEEE P1003.2 Draft 11.2 – September 1991

```
cat doc doc.end > doc
```

causes the original data in `doc` to be lost.

Due to changes made to subclause 2.11.4 in Draft 11, the description of the *file* operand now states that – must be accepted multiple times, as in historical practice. This allows the command:

```
cat start - middle - end > file
```

when standard input is a terminal, to get two arbitrary pieces of input from the terminal with a single invocation of `cat`. Note, however, that if standard input is a regular file, this would be equivalent to the command:

```
cat start - middle /dev/null end > file
```

because the entire contents of the file would be consumed by `cat` the first time – was used as a *file* operand and an end-of-file condition would be detected immediately when – was referenced the second time.

History of Decisions Made

None.

4.5 `cd` — Change working directory

4.5.1 Synopsis

`cd` [*directory*]

4.5.2 Description

The `cd` utility shall change the working directory of the current shell execution environment; see 3.12.

When invoked with no operands, and the **HOME** environment variable is set to a nonempty value, the directory named in the **HOME** environment variable shall become the new working directory. If **HOME** is empty or is undefined, the default behavior is implementation defined.

4.5.3 Options

None.

4.5.4 Operands

The following operands shall be supported by the implementation:

directory An absolute or relative pathname of the directory that becomes the new working directory. The interpretation of a relative pathname by `cd` depends on the **CDPATH** environment variable. If *directory* is –, the results are implementation defined.

4.5.5 External Influences

4.5.5.1 Standard Input

None.

4.5.5.2 Input Files

None.

4.5.5.3 Environment Variables

The following environment variables shall affect the execution of `cd`:

CDPATH A colon-separated list of pathnames that refer to directories. If the *directory* operand does not begin with a slash (/) character, and the first component is not dot or dot-dot, `cd` shall search for *directory* relative to each directory named in the **CDPATH** variable, in the order listed. The new working directory shall be set to the first matching directory found. An empty string in place of a directory pathname represents the current directory. If **CDPATH** is not set, it shall be treated as if it were an empty string.

HOME The name of the home directory, used when no *directory* operand is specified.

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

LC_MESSAGES This variable shall determine the language in which messages should be written.

4.5.5.4 Asynchronous Events

Default.

4.5.6 External Effects

4.5.6.1 Standard Output

If a nonempty directory name from **CDPATH** is used, an absolute pathname of the new working directory shall be written to the standard output as follows:

`"%s\n"`, *<new directory>*

Otherwise, there shall be no output.

4.5.6.2 Standard Error

Used only for diagnostic messages.

4.5.6.3 Output Files

None.

4.5.7 Extended Description

None.

4.5.8 Exit Status

The `cd` utility shall exit with one of the following values:

0 The directory was successfully changed.

>0 An error occurred.

4.5.9 Consequences of Errors

The working directory remains unchanged.

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

Examples, Usage

Editor's Note: A balloter requested that the following rationale be highlighted in 2
the D11.2 recirculation. 2

Since `cd` affects the current shell execution environment, it is generally provided
as a shell regular built-in. If it is called in a subshell or separate utility execution 1
environment, such as one of the following: 1

```
(cd /tmp)                                                    1
nohup cd                                                     1
find . -exec cd {} \;                                        1
```

it will not affect the working directory of the caller's environment. 1

The use of the **CDPATH** was introduced in the System V shell. Its use is analogous to the use of the **PATH** variable in the shell. Earlier systems such as the BSD C-shell used a shell parameter `cdpath` for this purpose.

History of Decisions Made

A common extension when **HOME** is undefined is to get the login directory from the user database for the invoking user. This does not occur on System V implementations.

Not included in this description are the features from the KornShell such as setting **OLDPWD**, toggling current and previous directory (`cd –`), and the two-operand form of `cd` (`cd` *old new*). This standard does not specify the results of `cd -` or of calls with more than one operand. Since these extensions are mostly used in interactive situations, they may be considered for inclusion in POSIX.2a. The result of `cd -` and of using no arguments with **HOME** unset or null have been made implementation defined at the request of the POSIX.6 security working group.

The setting of the **PWD** variable was removed from earlier drafts, as it can be replaced by `$(pwd)`.

4.6 `chgrp` — Change file group ownership

4.6.1 Synopsis

`chgrp [−R]` *group file* . . .

4.6.2 Description

The `chgrp` utility shall set the group ID of the file named by each *file* operand to the group ID specified by the *group* operand.

For each *file* operand, it shall perform actions equivalent to the POSIX.1 {8} *chown*() function, called with the following arguments:

(1) The *file* operand shall be used as the *path* argument.

(2) The user ID of the file shall be used as the *owner* argument.

(3) The specified *group ID* shall be used as the *group* argument.

4.6.3 Options

The `chgrp` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following option shall be supported by the implementation:

`−R` Recursively change file group IDs. For each *file* operand that names a directory, `chgrp` shall change the group of the directory and all files in the file hierarchy below it.

4.6.4 Operands

The following operands shall be supported by the implementation:

group A group name from the group database or a numeric group ID. Either specifies a group ID to be given to each file named by one of the *file* operands. If a numeric *group* operand exists in the group database as a group name, the group ID number associated with that group name is used as the group ID.

file A pathname of a file whose group ID is to be modified.