IEEE P1003.2a Draft 8 – December 1991

Section 4: Revisions to Execution Environment Utilities

4.12 `command` — Execute a simple command

⇒ **4.12.1 `command` Synopsis.** *Add the following lines to the* `command` *utility Synopsis:*

On systems supporting the User Portability Utilities Option, the following 7
forms shall also be supported: 7

`command [ −v | −V ]` *command_name*

⇒ **4.12.2 `command` Description.** *Add the following paragraph to the end of the* `command` *utility Description:*

On systems supporting the User Portability Utilities Option, the `command` utility shall also provide information concerning how a command name will be interpreted by the shell; see `−v` and `−V`.

⇒ **4.12.3 `command` Options.** *Add the following options in the correct sorted order:*

`−v` Write a string to standard output that indicates the pathname
or command that will be used by the shell, in the current shell 7
execution environment (see 3.12), to invoke *command_name* . 7

— Utilities, regular built-in utilities, *command_names* including a slash character, and any implementation-provided functions that are found using the **PATH** variable (as described in 3.9.1.1), shall be written as absolute pathnames.

— Shell functions, special built-in utilities, regular built-in 7
utilities not associated with a **PATH** search, and shell 7
reserved words shall be written as just their names. 7

— An alias shall be written as a command line that represents its alias definition.

— Otherwise, no output shall be written and the exit status shall reflect that the name was not found.

`-V` Write a string to standard output that indicates how the name given in the *command_name* operand will be interpreted by the shell, in the current shell execution environment (see 7
3.12). Although the format of this string is unspecified, it 7
shall indicate in which of the following categories *command_name* falls and include the information stated:

— Utilities, regular built-in utilities, and any implementation-provided functions that are found using the **PATH** variable (as described in 3.9.1.1), shall be identified as such and include the absolute pathname in the string.

— Other shell functions shall be identified as functions.

— Aliases shall be identified as aliases and their definitions shall be included in the string.

— Special built-in utilities shall be identified as special built-in utilities.

— Regular built-in utilities not associated with a **PATH** 7
search shall be identified as regular built-in utilities. (The 7
term ''regular'' need not be used.) 7

— Shell reserved words shall be identified as reserved words.

⇒ **4.12.6.1 `command` Standard Output.** *Change* 'None. '' *to:*

When the `-v` option is specified, standard output shall be formatted as:

`"%s\n"`, <*pathname or command*>

When the `-V` option is specified, standard output shall be formatted as:

`"%s\n"`, <*unspecified*>

⇒ **4.12.8 `command` Exit Status.** *Change the first sentence, beginning* 'The command utility shall exit ... '' *to*

When the `-v` or `-V` options are specified, the `command` utility shall exit with 8
one of the following values:

0 Successful completion.

>0 The *command_name* could not be found or an error occurred.

Otherwise, the `command` utility shall exit with one of the following values:

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

The `command` –`v` and –`V` options were added to the UPE to satisfy requirements from users that are currently accomplished by three different historical utilities: `type` in the System V shell, `whence` in the KornShell, and `which` in the C-shell. Since there is no historical agreement on how and what to accomplish here, the POSIX `command` utility was enhanced and the historical utilities were left unmodified. The C-shell `which` merely conducts a path search. The KornShell `whence` is more elaborate—in addition to the categories required by POSIX, it also reports on tracked aliases, exported aliases, and undefined functions.

Since the –`v` and –`V` options of `command` produce output in relation to the current 7
shell execution environment, `command` is generally provided as a shell regular 7
built-in. If it is called in a subshell or separate utility execution environment, 7
such as one of the following: 7

```
(PATH=foo command -v)      7
nohup command -v           7
```

it will not necessarily produce correct results. For example, when called with 7
`nohup` or an *exec* function, in a separate utility execution environment, most 7
implementations will not be able to identify aliases, functions, or special built-ins. 7

Two types of regular built-ins could be encountered on a system and these are 7
described separately by `command`. The description of command search in 3.9.1.1 7
allows for a standard utility to be implemented as a regular built-in as long as it 7
is found in the appropriate place in a **PATH** search. So, for example, 7
`command -v true` might yield `/bin/true` or some similar pathname. Other 7
implementation-provided utilities that are not defined by POSIX.2 might exist only 7
as built-ins and have no pathname associated with them. These will produce out- 7
put identified as (regular) built-ins. Applications encountering these will not be 7
able to count on *exec*ing them, using them with `nohup`, overriding them with a 7
different **PATH**, etc. 7

The output format of –`V` was left mostly unspecified because human users are its only audience. Applications should not be written to care about this information; they can use the output of –`v` to differentiate between various types of commands, but the additional information that may be emitted by the more verbose –`V` is not needed and should not be arbitrarily constrained in its verbosity or localization for application parsing reasons.

4.32 `kill` — Terminate or signal processes

⇒ **4.32.4 `kill` Operands.** *In clause 4.32, modify the description of the pid operand in the Operands subclause:*

pid One of the following:

(1) *<current contents>*

(2) On systems supporting both the POSIX.1 {8} job control option and the User Portability Utilities Option, a job control job ID (see 2.2.2.206) that identifies a background process group to be signaled. The job control job ID notation is applicable only for invocations of `kill` in the current shell execution environment; see 3.12.

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

The job control job ID notation is not required to work as expected when `kill` is operating in its own utility execution environment. In either of the following examples:

```
nohup kill %1 &
system("kill %1");
```

the `kill` operates in a different environment and will not share the shell's understanding of job numbers.