IEEE P1003.2a Draft 8 – December 1991

5.13 `fg` — Run jobs in the foreground

This utility shall be provided on systems that support both the User Portability 7
Utilities Option and the POSIX.1 {8} job control option. On other systems it is 7
optional. 7

5.13.1 Synopsis

`fg` [*job_id*]

5.13.2 Description

If job control is enabled (see the description of `set -m` in 3.14.11), the `fg` utility shall move a background job from the current environment (see 3.12) into the foreground.

Using `fg` to place a job into the foreground shall remove its process ID from the 7
list of those ‘‘known in the current shell execution environment’’; see 3.9.3.1.

5.13.3 Options

None.

5.13.4 Operands

The following operand shall be supported by the implementation:

job_id Specify the job to be run as a foreground job. If no *job_id* operand is given, the *job_id* for the job that was most recently suspended, placed in background, or run as a background job shall be used. The format of *job_id* is described in 2.2.2.206.

5.13.5 External Influences

5.13.5.1 Standard Input

None.

5.13.5.2 Input Files

None.

5.13.5.3 Environment Variables

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

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.

5.13.5.4 Asynchronous Events

Default.

5.13.6 External Effects

5.13.6.1 Standard Output

The `fg` utility shall write the command line of the job to standard output in the following format:

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

5.13.6.2 Standard Error

Used only for diagnostic messages.

5.13.6.3 Output Files

None.

5.13.7 Extended Description

None.

5.13.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.13.9 Consequences of Errors

If job control is disabled, the `fg` utility shall exit with an error and no job shall be placed in the foreground.

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

Usage, Examples

The `fg` utility will not work as expected when it is operating in its own utility execution environment because that environment will have no applicable jobs to manipulate. See the rationale for `bg`. For this reason, `fg` is generally implemented as a shell regular built-in.

History of Decisions Made

The extensions to the shell specified in POSIX.2 and POSIX.2a have mostly been based on features provided by the KornShell. The job control features provided by `bg`, `fg`, and `jobs` are also based on the KornShell. The developers of the standard examined the characteristics of the C-Shell versions of these utilities and found that differences exist. Despite widespread use of the C-Shell, the KornShell versions were selected for this standard to maintain a degree of uniformity with the rest of the KornShell features selected (such as the very popular command-line editing features).

5.14 `file` — Determine file type

5.14.1 Synopsis

`file` *file* . . .

5.14.2 Description

The `file` utility shall perform a series of tests on each specified *file* in an attempt to classify it:

(1) If the file is not a regular file, its file type shall be identified. The file types directory, FIFO, block special, and character special shall be identified as such. Other implementation-defined file types may also be identified.

(2) If the file is a regular file, and:

  (a) The file is zero-length, it shall be identified as an empty file.

  (b) The file is not zero-length, `file` shall examine an initial segment of the file and shall make a guess at identifying its contents or whether it is an executable binary file. (The answer is not guaranteed to be correct.)

If *file* does not exist, cannot be read, or its file status could not be determined, the output shall indicate that the file was processed, but that its type could not be determined.

5.14.3 Options

None.

5.14.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a file to be tested.

5.14.5 External Influences

5.14.5.1 Standard Input

None.