IEEE P1003.2a Draft 8 – December 1991

History of Decisions Made

None.

5.12 `fc` — Process command history list

5.12.1 Synopsis

`fc` [`−e` *editor*] [*first* [*last*]]

`fc −l` [`−nr`] [*first* [*last*]]

`fc −s` [*old=new*] [*first*]

5.12.2 Description

The `fc` utility shall list, or shall edit and re-execute, commands previously entered to an interactive `sh` (see 4.56).

The command history list shall reference commands by number. The first number in the list is selected arbitrarily. The relationship of a number to its command shall not change except when the user logs in and no other process is accessing the list, at which time the system may reset the numbering to start the oldest retained command at another number (usually 1). When the number reaches an implementation-defined upper limit, which shall be no smaller than the value in **HISTSIZE** or 32 767 (whichever is greater), the shell may wrap the numbers, starting the next command with a lower number (usually 1). However, despite this optional wrapping of numbers, `fc` shall maintain the time-ordering sequence of the commands. For example, if four commands in sequence are given the numbers 32 766, 32 767, 1 (wrapped), and 2 as they are executed, command 32 767 is considered the command previous to 1, even though its number is higher.

When commands are edited (when the `−l` option is not specified), the resulting line(s) shall be entered at the end of the history list and then re-executed by `sh`. The `fc` command that caused the editing shall not be entered into the history list. If the editor returns a nonzero exit status, this shall suppress the entry into the history list and the command re-execution. Any command line variable assignments or redirection operators used with `fc` shall affect both the `fc` command itself as well as the command that results. For example,

```
fc -s -1 2>/dev/null
```

reinvokes the previous command, suppressing standard error for both `fc` and the previous command.

5.12.3 Options

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

The following options shall be supported by the implementation:

`−e` *editor* Use the editor named by *editor* to edit the commands. The *editor* string is a utility name, subject to search via the **PATH** variable (see 2.6). The value in the **FCEDIT** variable shall be used as a default when `−e` is not specified. If **FCEDIT** is null or unset, `ed` shall be used as the editor.

`−l` (The letter ell.) List the commands rather than invoking an edi-
tor on them. The commands shall be written in the sequence 8
indicated by the *first* and *last* operands, as affected by `−r`, with 8
each command preceded by the command number. 8

`−n` Suppress command numbers when listing with `−l`.

`−r` Reverse the order of the commands listed (with `−l`) or edited 8
(with neither `−l` nor `−s`). 8

`−s` Re-execute the command without invoking an editor. 8

5.12.4 Operands

The following operands shall be supported by the implementation:

first
last Select the commands to list or edit. The number of previous commands that can be accessed shall be determined by the value of the **HISTSIZE** variable. The value of *first* or *last* or both shall be one of the following:

[+]*number* A positive number representing a command number; command numbers can be displayed
with the `−l` option. 8

−*number* A negative decimal number representing the command that was executed *number* of commands
previously. For example, −1 is the immediately 8
previous command. 8

string A string indicating the most recently entered command that begins with that string. If the *old*=*new* operand is not also specified with `−s`, the string form of the *first* operand cannot contain an embedded equal sign.

When the synopsis form with `−s` is used: 7

— If *first* is omitted, the previous command shall be used. 7

For the synopsis forms without `-s`: 7

— If *last* is omitted, *last* shall default to *first* .

— If *first* and *last* are both omitted, the previous sixteen commands shall be listed or the previous one command shall be edited (based on the `-l` option).

— If *first* and *last* are both present, all of the commands from 8
first to *last* shall be edited (without `-l`) or listed (with `-l`). 8
Editing multiple commands shall be accomplished by present- 8
ing to the editor all of the commands at one time, each com- 8
mand starting on a new line. If *first* represents a newer com- 8
mand than *last* , the commands shall be listed or edited in 8
reverse sequence, equivalent to using `-r`. For example, the 8
following pairs of commands are equivalent: 8

```
fc -r 10 20          fc    30 40
fc    20 10          fc -r 40 30
```
8
8

— When a range of commands is used, it shall not be an error to 8
specify *first* or *last* values that are not in the history list; `fc` 8
shall substitute the value representing the oldest or newest 8
command in the list, as appropriate. For example, if there are 8
only ten commands in the history list, numbered 1 to 10, 8

```
fc -l
fc 1 99
```
8
8

shall list and edit, respectively, all ten commands. 8

old=new Replace the first occurrence of string *old* in the commands to be 8
re-executed by the string *new* . 8

5.12.5 External Influences

5.12.5.1 Standard Input

None.

5.12.5.2 Input Files

None.

5.12.5.3 Environment Variables

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

FCEDIT This variable, when expanded by the shell, shall be interpreted as the default value for the `-e` *editor* option's *editor* option-argument. If **FCEDIT** is null or unset, `ed` shall be used as the editor.

HISTFILE This variable shall be interpreted as a pathname naming a command history file. If the **HISTFILE** variable is not set, the shell may attempt to access or create a file `.sh_history` in the user's home directory. If the shell
cannot obtain both read and write access to, or create, the 8
history file, it it shall use an unspecified mechanism that will allow the history to operate properly. Implementations may choose to disable the history list mechanism for users with appropriate privileges who do not set **HISTFILE** ; the specific circumstances under which this will occur are implementation defined. If more than one instance of the shell is using the same history file, it is unspecified how updates to the history file from those shells interact. As entries are deleted from the history file, they shall be deleted oldest first. It is unspecified when history file entries are physically removed from the history file.

HISTSIZE This variable shall be interpreted as a decimal number representing the limit to the number of previous com-
mands that are accessible. If this variable is unset, an 7
unspecified default ≥ 128 shall be used. The maximum 7
number of commands in the history list is unspecified, but
shall be at least 128. An implementation may choose to 8
access this variable only when initializing the history file. 8
Therefore, it is unspecified whether changes made to 8
HISTSIZE after the history file has been initialized are 8
effective. 8

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 and input files).

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

5.12.5.4 Asynchronous Events

Default.

5.12.6 External Effects

5.12.6.1 Standard Output

When the `−l` option is used to list commands, the format of each command in the list shall be as follows:

`"%d\t%s\n"`, *<line number>* , *<command>*

If both the `−l` and `−n` options are specified, the format of each command shall be:

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

If the *<command>* consists of more than one line, the lines after the first shall be displayed as:

`"\t%s\n"`, *<continued-command>*

5.12.6.2 Standard Error

Used only for diagnostic messages.

5.12.6.3 Output Files

None.

5.12.7 Extended Description

None.

5.12.8 Exit Status

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

0 Successful completion of the listing.

>0 An error occurred.

Otherwise, the exit status shall be that of the commands executed by `fc`.

5.12.9 Consequences of Errors

Default.

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

Usage, Examples

Command numbers have no ordering effects; they are like serial numbers. The `−r` option and *−number* operand address the sequence of command execution, regardless of serial numbers. So, for example, if the command number wrapped back to 1 at some arbitrary point, there would be no ambiguity associated with traversing the wrap point. For example, if the command history were:

```
32766: echo 1
32767: echo 2
1: echo 3
```

the number `-2` refers to command 32767 because it is the second previous, regardless of serial number.

Since editors sometime use file descriptors as integral parts of their editing, redirecting their file descriptors as part of the `fc` command can produce unexpected results. For example, if `vi` is the **FCEDIT** editor, the command

```
fc -s | more
```

will not work correctly on many systems.

History of Decisions Made

This utility is based on the KornShell's `fc` built-in.

An earlier draft specified the `−e` option as `[-e editor [old=new]]`, which is not existing practice. Existing practice in `fc` of `[-e editor]` or `[-e - [old=new]]` is acceptable, but not together. To clarify this, a new option `−s` was introduced replacing the `[-e -]`. This resolves the conflict and makes `fc` conform to the Utility Syntax Guidelines.

HISTFILE Users on windowing systems may want to have separate history files for each window by setting **HISTFILE** as follows:

```
HISTFILE=$HOME/.sh_hist$$
```

Some implementations of the KornShell check for the super-user and do not create a history file unless **HISTFILE** is set. This is done primarily to avoid creating unlinked files in the root file system when logging in during single-user mode. **HISTFILE** must be set for the super-user to have history.

HISTSIZE Needed to limit the size of history files. It is the intent of this standard that when two shells share the same history file, commands that are entered in one shell shall be accessible by the other shell. Because of the difficulties of synchronization over a network, the exact nature of the interaction is unspecified.

Consideration was given to omitting the `fc` utility in favor of the command-line 8
editing feature in `sh`. For example, in `vi` editing mode, typing `<ESC>v` is 8
equivalent to 8

```
EDITOR=vi fc
```
8

However, the `fc` utility allows the user the flexibility to edit multiple commands 8
simultaneously (such as `fc 10 20`) and to use editors other than those supported 8
by `sh` for command-line editing. 8

In the KornShell, the alias `r` (''re-do'') is preset to `fc -e -` (equivalent to the 8
POSIX `fc –s`). This is probably an easier command name to remember than `fc` 8
(''fix command''), but it does not meet the Utility Syntax Guidelines. Renaming 8
`fc` to `hist` or `redo` was considered, but since this description closely matches his- 8
torical KornShell practice already, such a renaming was seen as gratuitous. 8
Users are free to create aliases whenever odd historical names such as `fc`, `awk`, 8
`cat`, `grep`, or `yacc` are standardized by POSIX. 8

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.