IEEE P1003.2a Draft 8 – December 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

5.10.7.5.21 `showmatch, sm`

[Default *off*]

If `showmatch` is set, in visual mode, when a `)` or `}` is typed, the matching `(` or `{` shall be shown if it is still on the screen. This editor option need not be supported on block-mode terminals.

5.10.7.5.22 `showmode`

[Default *off*]

If `showmode` is set, in visual mode, the current mode that the editor is in shall be written on the last line of the screen. Modes that shall be reported are command mode and input mode; other unspecified modes may be written.

5.10.7.5.23 `tabstop, ts`

[Default 8]

The value of this option shall specify the software tab stops to be used by the editor to expand tabs in the input.

5.10.7.5.24 `tags`

[Default *see text*] 7

The value of this option can be a string representing `<space>`-separated pathnames that shall be used as tag files for the `tag` command. A requested tag shall be searched for sequentially in the specified files. By default, filenames of `tags` shall be searched for in the current directory and in other implementation-defined directories.

5.10.7.5.25 `term`

[Default from the environment **TERM**]

The value of this option can be a string representing the terminal type of the output device. The default shall be taken from the **TERM** variable in the environment; see 5.10.5.3 for default values for **TERM**.

5.10.7.5.26 `terse`

[Default *off*]

If `terse` is set, error messages may be less verbose. However, except for this caveat, error messages are unspecified. Furthermore, not all error messages need change for different settings of this option.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.10.7.5.27 `warn`

[Default *on*]

If `warn` is set, `ex` shall write a warning message to standard error if the contents of the buffer have not been saved before a ! command escape.

5.10.7.5.28 `window`

[Default *see text*]

The value of this option shall determine the default number of lines in a screenful, as written by the `z` command. When in visual mode, the number of lines output when moving up or down the file by a *screenful* (see 5.35). The value of `window` can be unrelated to the real screen size, except that it shall be set on entry to
be the current number of screen lines. (The current number of screen lines shall 7
be determined by the system or overridden by the user, as described for **LINES** in 7
5.10.5.3 and 2.6.) The baud rate of the terminal line may reduce the default in an 7
implementation-defined manner. The default value of `windows` also can be over- 7
ridden by specifying a window size using the `-w` command-line option. 7

5.10.7.5.29 `wrapscan, ws`

[Default *on*]

If `wrapscan` is set, searches (using `//` or `??`) shall wrap around the end of the editing buffer; when unset, searches shall stop at the beginning of the editing buffer for `??`, or at the end of the editing buffer for `//`.

5.10.7.5.30 `wrapmargin, wm`

[Default 0]

If the value of this option is greater than zero (say *n*) in visual mode during text
entry, then, in the POSIX Locale, a `<newline>` shall replace all consecutive 7
`<blank>`s, at the boundary between a `<blank>` and a non-`<blank>`, so that lines 7
shall end at least *n* spaces from the ending margin of the terminal screen. (The 7
ending margin shall be determined by the system or overridden by the user, as 7
described for **COLUMNS** in 5.10.5.3 and 2.6.) If a line consists of a sequence of 7
`<non->`blank characters long enough such that it extends continuously from the beginning margin to beyond the ending margin, that sequence shall not be broken by the action of this option. If the value is zero, no wrapping shall be performed.

5.10.7.5.31 `writeany, wa`

[Default *off*]

If `writeany` is set, file-overwriting checks shall be inhibited that would otherwise 7
be made before `write` and `xit` commands, or before an automatic write (see editor option `autowrite`), allowing a write to any file (provided permissions allow it).

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.10.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.10.9 Consequences of Errors

When an error in the input script is encountered, or when an error is detected 7
that is a consequence of the data (not) present in the file or due to an external 7
condition such as a read or write error: 7

— If the standard input is a terminal device file, all input shall be flushed, 8
and a new command read. 8

— If the standard input is a regular file, `ex` shall terminate with a nonzero 7
exit status. 7

— If the command in error was one of those specified with the command-line 7
`−c` option (or the obsolescent `+` option), all of the remaining `−c` commands 7
shall be flushed, and a new command read from standard input. 7

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

Usage, Examples

All of the following are valid *addresses* :

`+++` three lines after the current line

`/`*re*`/-` one line before the next occurrence of *re*

`-2` two lines before the current line

The rules for case-insensitive matching (editor option `ignorecase`) may lead to counter-intuitive situations when uppercase characters are used in range expressions. Historical versions of this text implied that the process was:

(1) Take a line of text from buffer

(2) Convert uppercase to lowercase in text line

(3) Convert uppercase to lowercase in RE, except in character class 7
specifications 7

(4) Match RE against text

This would mean that, with `ignorecase` in effect, the text

```
The cat sat on the mat
```

would be matched by

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

```
/ˆthe/
```

but not by

```
/ˆ[A-Z]he/
```

If a SIGSEGV signal is received while ex is saving a file, the file might not be successfully saved.

The next command can accept more than one file, so usage such as

```
next ‘ls [abc]*‘
```

is valid; it would not be valid for the edit or read commands, for example, because they expect only one file and unspecified results occur.

An example of case conversion with the s command:

```
:p
The cat sat on the mat.
:s/\<.at\>/\u&/gp
The Cat Sat on the Mat.
:s/S\(.*\)M/S\U\1\eM/p
The Cat SAT ON THE Mat.
```

The following table is a condensed version of information contained in the normative text. It is presented here to facilitate the review for accuracy of the ex commands and the editor options that affect those commands.

ex Command	Editor Option	Description
a[ppend]	autoindent	Determines starting indentation by the line immediately preceding the append.
a[ppend]!	autoindent	a! causes the autoindent editor option setting to be toggled for the duration of this command only.
c[hange]	autoindent	Determines starting indentation based on the first line that was changed.
c[hange]!	autoindent	c! causes the autoindent editor option setting to be toggled for the duration of this command only.
e[dit] ex	autowrite	Causes the buffer to be written to the current file if it has been modified. (e! or ex! causes write not to occur.)
	writeany	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
g[lobal] v	autoindent	Editor option is inhibited when using this command.
	autoprint	Editor option is inhibited when using this command.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

	`ignorecase`	Characters that have uppercase and lowercase representations shall have those representations considered as equivalent for purposes of regular expression comparison.
	`magic`	When set, the characters `^ $ . * [ ~` have special meaning in regular expressions. When unset, only characters `^ $` have special meaning in regular expressions.
	`report`	Editor option is inhibited when using this command.
`i[nsert]`	`autoindent`	Determines starting indentation based on the line that is inserted before.
`i[nsert]!`	`autoindent`	`i!` causes the autoindent editor option setting to be toggled for the duration of this command only.
`n[ext]`	`autowrite`	Causes the buffer to be written to the current file if it has been modified. (`n!` causes write not to occur.)
	`writeany`	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
`p[rint]`	`list`	Causes nonprintable characters to be written, 7 and marks the end of line with $. 7
	`number`	Causes lines to be written with a line number before the text.
`rew[ind]`	`autowrite`	Causes the buffer to be written to the current file if it has been modified. (`rew!` causes write not to occur.)
	`writeany`	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
`sh[ell]`	`shell`	The pathname of the shell to be invoked.
`s[ubstitute]`	`ignorecase`	Characters that have uppercase and lowercase representations shall have those representations considered as equivalent for purposes of regular expression comparison.
	`magic`	When set, the characters `^ $ . * [ ~` have special meaning in regular expressions. When unset, only characters `^ $` have special meaning in regular expressions.
	`report`	Determines the number of lines that can be changed by an editor command before a report is generated.
	`wrapscan`	When set, searches (using `//` or `??`) shall wrap

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

		around the end of the file. When unset, searches shall stop at the beginning of the file for `??`, or at the end of the file for `//`.
`su[spend]` `st[op]`	`autowrite`	Causes the buffer to be written to the current file if it has been modified. (`su!` or `st!` causes write not to occur.)
`ta[g]`	`autowrite`	Causes the buffer to be written to the current file if it has been modified. (`ta!` causes write not to occur.)
	`tags`	File searched when looking for a tag.
	`writeany`	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
`vi[sual]`	`window`	Default window size used when invoking `vi` command.
`w[rite]`	`readonly`	Prohibits write to file unless w! is entered.
	`writeany`	Inhibits checks before a write, allowing a write to any file (provided that permissions allow it).
`x[it]`	`readonly`	Prohibits write to file unless `x!` is entered.
	`writeany`	Inhibits checks before an exit, allowing a write to any file (provided that permissions allow it).
`z`	`window`	Default screen size if not explicitly stated.
`!` (Escape)	`autowrite`	Causes the buffer to be written to the current file if it has been modified.
	`shell`	The pathname of the shell to be invoked.
	`warn`	Warn user before a ! command escape if there has been ‘‘[No write since last change]’’
	`writeany`	Inhibits checks before an automatic write, allowing a write to any file (provided that permissions allow it).
`<`	`shiftwidth`	Determines the number of character positions to be shifted.
`>`	`shiftwidth`	Determines the number of character positions to be shifted.
eof	`scroll`	Determines the number of lines to scroll.

The `−s` option (and its obsolescent single-hyphen form) suppresses all interactive 8
user feedback and is useful for editing scripts in batch jobs. The list of specific 8
effects is derived from historical practice. The terminal type ‘‘incapable of sup- 8
porting visual mode’’ historically has been named `dumb`. 8

The uses described for `control-V` can also be accomplished with `control-Q`, 8
which is useful on terminals that use `control-V` for the down-arrow function. 8
However, most historical implementations use `control-Q` for the *termios* START 8
character, so the editor will generally not receive the `control-Q` unless `stty` 8
`−ixon` mode is set. Any of the command characters described in this standard 8

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

can be made ineffective by their selection as *termios* control characters, using the 8
`stty` utility or other methods described in POSIX.1 {8}. 8

History of Decisions Made

It is recognized that `ex` visual mode (`vi`) and related features would be difficult, if not impossible, to implement satisfactorily on a synchronous terminal, thus it is not a mandatory requirement that such features should work on all terminals. It is the intention, however, that an `ex` implementation should provide the full set of capabilities on all terminals capable of supporting them.

A conforming syntax has been provided for `ex`.

A ''restricted editor'' (both the historical `red` utility and modifications to `ex`) were considered and rejected for inclusion. Neither option provided the level of security that many users might expect.

The `−c` replacement for ''+command'' was inspired by the `−e` option of `sed`.

The `−l` command for LISP mode was omitted as it was considered that it would be difficult to justify the inclusion of such programming language-dependent features for one language and not for others. Similarly the `lisp` edit mode option was omitted.

The `−t` option was introduced because the `ctags` command appears in POSIX.2 and the option is already widely available.

The `−x` option for encryption, and the associated `crypt` utility, were omitted because the algorithm used was not specifiable and the export control laws of some nations make it difficult to export cryptographic technology.

It was agreed that the ability to edit files of arbitrary type can be occasionally useful, but it was not felt necessary to mandate that an `ex` implementation be required to handle other than text files.

The paragraph in Input Files, ''By default, . . . ,'' is intended to close a long-standing security hole in `ex` and `vi`, that of the little-known ''modelines.'' This ''feature'' is that whereby any line in the first or last five lines of the file containing the strings `"ex:"` or `"vi:"` (and, apparently `"ei:"` or `"vx:"`) is considered to be a line containing editor commands and `ex` interprets all the text up to the next `:` or `<newline>` as a command. Consider the consequences, for example, of an unsuspecting user using `ex` or `vi` as their editor when replying to a mail message in which a line such as `ex:! rm −rf *:` appeared in the signature lines. It was strongly felt that an editor should not *by default* interpret any lines of a file. If this feature is desired it should be under the user's control either by the use of a command-line option or by placing some control line in a initialization file. This approach has indeed already been adopted by some vendors (HP-UX has the `modelines` option, for instance, which is turned off by default).

The current documentation for `ex` seems to imply a particular implementation of a scheme for recovering files after some ''disaster.'' Such details were eliminated.

Explicit dependence on the ASCII character set has been avoided where possible, hence the use of an ''implementation-defined multicharacter sequence'' for the

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

display of nonprintable characters in preference to the historical usage of, say, `^I` for `<tab>`. Room is left for an implementation to select some other means of displaying characters unambiguously: combinations of octal or hexadecimal values with leading slashes along with the escape sequences in Table 2-13. (In fact, the historical `^I` is not really unambiguous at all, because it could actually be the characters `^` and `I`; allowing this historical practice is why the word ‘‘should’’ is used.) It is recommended that if an implementation chooses a new method for this writing that it match the formats used by `ed` and `sed` in POSIX.2.

The writing of NUL by the `p` command is not specified as a special case because the group did not want to require `ex` to support NUL characters in files and buffers.

The action of the `join` command in relation to its action with white space is only defined for the POSIX Locale as the correct amount of white space after a period varies; in Japanese none is required, in French only a single space, and so on.

References to ‘‘function keys’’ in the `map` command were omitted because the specification was too simple to be generally useful in a portable manner.

The `version` command was not included because no sufficiently general specification of the version information could be devised that would still be useful to a portable user. This command name should be used by suppliers who wish to provide version information about the `ex` and `vi` editors.

The `directory` editor option was omitted because it was implementation-specific.

The `edcompatible` editor option was omitted because it was not (`ed`-compatible).

The default for editor option `exrc` was reversed from common practice because it was perceived as a security hole by both the POSIX.2 and POSIX.6 working groups. The POSIX.2 group does not believe this to be an undue burden because those people who use `.exrc` file in directories other than `$HOME` can retain the old behavior (by setting this option in their **EXINIT** environment variables or their `$HOME/.exrc` files). Note that `noexrc` disables the use of `./.exrc`; `exrc` enables it. The default is to disable it.

The `paragraph` and `section` options are in the standard, even though their historical defaults are very oriented to the UNIX system `troff` text formatter, because a ‘‘portable user’’ could use the `{`, `}`, `[[`, or `]]` commands in visual mode and have the cursor stop in unexpected places.

The `redraw` and `slowopen` options were omitted because they are implementation dependent.

In historical implementations the default for editor option `window` is arrived at by the following steps:

(1) The initial default is half the number of lines on the terminal screen as defined in a terminal database (typically, *terminfo* or *termcap*).

(2) If dynamic screen size information is available either from environment variables (System V) or the TTY driver (BSD), this information is used

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

instead.

(3) If the number of lines is erroneous, the default is set to 23.

(4) If the baud rate is less than 2400, the default is set to 16.

(5) If the baud rate is less than 1200, the default is set to 8. This default can then be changed by the editor options below:

`window`	set the window size explicitly and always
`w300`	set `window` if baud rate is less than 1200
`w1200`	set `window` if baud rate is less than 2400
`w9600`	set `window` if baud rate is greater than 2400

The `w300`, `w1200`, and `w9600` options do not appear in this standard because of their lack of documentation and dependence on specific baud rates.

The action taken for signals other than SIGINT, SIGCONT, and SIGHUP is 7
unspecified because some implementations attempt to save the editing buffer in a 7
useful state when other signals are received. 7

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.11 expand — Convert tabs to spaces

5.11.1 Synopsis

`expand` [−t *tablist*] [*file* . . .]

Obsolescent Version :

`expand` [-*tabstop*] [-*tab1* , *tab2* , . . . , *tabn*] [*file* . . .]

5.11.2 Description

The `expand` utility shall write files or the standard input to the standard output with <tab> characters replaced with one or more <space> characters needed to pad to the next tab stop. Any <backspace> characters shall be copied to the output and cause the column position count for tab stop calculations to be decremented; the column position count shall not be decremented below zero.

5.11.3 Options

The `expand` utility shall conform to the utility argument syntax guidelines described in 2.10.2; the obsolescent version does not.

The following options shall be supported by the implementation:

−t *tablist* Specify the tab stops. The argument *tablist* shall consist of a single positive decimal integer or multiple positive decimal integers, separated by <blank>s or commas, in ascending order. If a single number is given, tabs shall be set *tablist* column positions apart instead of the default 8. If multiple numbers are given, the tabs shall be set at those specific column positions.

Each tab-stop position *N* shall be an integer value greater than zero, and the list shall be in strictly ascending order. This shall be taken to mean that, from the start of a line of output, tabbing to position *N* shall cause the next character output to be in the (N+1)th column position on that line.

In the event of `expand` having to process a <tab> character at a position beyond the last of those specified in a multiple tab-stop list, the <tab> shall be replaced by a single <space> in the output.

In the obsolescent version, the single number is specified as *tabstop* with a leading minus; multiple tab stops are specified after a leading minus as *tab1* , *tab2* , etc.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.11.4 Operands

The following operand shall be supported by the implementation:

file The pathname of a text file to be used as input.

5.11.5 External Influences

5.11.5.1 Standard Input

See Input Files.

5.11.5.2 Input Files

Input files shall be text files.

5.11.5.3 Environment Variables

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

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 interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files), the processing of `<tab>`s and `<space>`s, and for the determination of the width in column positions each character would occupy on a constant-width-font output device.

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

5.11.5.4 Asynchronous Events

Default.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.11.6 External Effects

5.11.6.1 Standard Output

The standard output shall be equivalent to the input files with <tab> characters converted into <space>s, as specified by 5.11.2.

5.11.6.2 Standard Error

Used only for diagnostic messages.

5.11.6.3 Output Files

None.

5.11.7 Extended Description

None.

5.11.8 Exit Status

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

- 0 Successful completion.
- >0 An error occurred.

5.11.9 Consequences of Errors

The `expand` utility shall terminate with an error message and nonzero exit status upon encountering difficulties accessing one of the *file* operands.

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

Usage, Examples

The `expand` utility is useful for preprocessing text files (before sorting, looking at specific columns, etc.) that contain <tab>s.

See the definition of *column position* in 2.2.2.31.

The *tablist* option-argument consists of integers in ascending order. Utility Syntax Guideline 8 (see 2.10.2) mandates that `expand` shall accept the integers (within the single argument) separated using either commas or <blank>s.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 editor on them. The commands shall be written in the sequence indicated by the *first* and *last* operands, as affected by `−r`, with 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 (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 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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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

For the synopsis forms without `-s`:

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

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

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

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

shall list and edit, respectively, all ten commands.

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

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`:

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 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 commands that are accessible. If this variable is unset, an unspecified default ≥ 128 shall be used. The maximum number of commands in the history list is unspecified, but shall be at least 128. An implementation may choose to access this variable only when initializing the history file. Therefore, it is unspecified whether changes made to **HISTSIZE** after the history file has been initialized are effective.

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).

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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`.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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` 8
option and *–number* operand address the sequence of command execution, regard- 8
less of serial numbers. So, for example, if the command number wrapped back to 8
1 at some arbitrary point, there would be no ambiguity associated with traversing 8
the wrap point. For example, if the command history were: 8

```
32766: echo 1
32767: echo 2
1: echo 3
```
8 8 8

the number `-2` refers to command 32767 because it is the second previous, regard- 8
less of serial number. 8

Since editors sometime use file descriptors as integral parts of their editing, 8
redirecting their file descriptors as part of the `fc` command can produce unex- 8
pected results. For example, if `vi` is the **FCEDIT** editor, the command 8

```
fc -s | more
```
8

will not work correctly on many systems. 8

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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 editing feature in `sh`. For example, in `vi` editing mode, typing `<ESC>v` is equivalent to

```
EDITOR=vi fc
```

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

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

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

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.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.