Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'fc.1p' command

$ man fc.1p

FC(1P) POSIX Programmer's Manual FC(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

fc ? process the command history list

SYNOPSIS

fc [-r] [-e editor] [first [last]]

fc -l [-nr] [first [last]]

fc -s [old=new] [first]

DESCRIPTION

The fc utility shall list, or shall edit and re-execute, commands pre? viously entered to an interactive sh.

The command history list shall reference commands by number. The first number in the list is selected arbitrarily. The relationship of a num? ber 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 32767 (whichever is greater), the shell may wrap the numbers, starting the

next command with a lower number (usually 1). However, despite this op? tional wrapping of numbers, fc shall maintain the time-ordering se? quence of the commands. For example, if four commands in sequence are given the numbers 32766, 32767, 1 (wrapped), and 2 as they are exe? cuted, command 32767 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 re? sulting lines 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 non-zero exit status, this shall suppress the entry into the history list and the command re-execution. Any command line variable assignments or redi? rection 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.

OPTIONS

The fc utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

-e editor Use the editor named by editor to edit the commands. The edi? tor string is a utility name, subject to search via the PATH variable (see the Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables). 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.

-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).

-s Re-execute the command without invoking an editor.

OPERANDS

The following operands shall be supported:

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 op? tion.

-number A negative decimal number representing the command that was executed number of commands previously. For example, -1 is the immediately previous com? mand.

string A string indicating the most recently entered com? mand 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 embed? ded <equals-sign>.

When the synopsis form with -s is used:

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

For the synopsis forms without -s:

* If last is omitted, last shall default to the previous command when -l is specified; otherwise, it shall default to first.

* If first and last are both omitted, the previous 16 com? mands shall be listed or the previous single 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 accom?

plished 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 commands on the first line are equivalent to the corresponding commands on the second:

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 er? ror 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 his? tory 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.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of fc:

FCEDIT This variable, when expanded by the shell, shall determine 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 Determine 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 directory referred to by

the HOME environment variable. If the shell cannot obtain both read and write access to, or create, the history file, it shall use an unspecified mechanism that allows the history to operate properly. (References to history ``file'' in this section shall be understood to mean this unspecified mecha? nism in such cases.) An implementation may choose to access this variable only when initializing the history file; this initialization shall occur when fc or sh first attempt to re? trieve entries from, or add entries to, the file, as the re? sult of commands issued by the user, the file named by the ENV variable, or implementation-defined system start-up files. In some historical shells, the history file is ini? tialized just after the ENV file has been processed. There? fore, it is implementation-defined whether changes made to HISTFILE after the history file has been initialized are ef? fective. 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 occurs are implementation-defined. If more than one instance of the shell is using the same history file, it is unspeci? fied how updates to the history file from those shells inter? act. 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 Determine a decimal number representing the limit to the num? ber of previous commands that are accessible. If this vari? able is unset, an unspecified default greater than or equal to 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, as described under HISTFILE. Therefore, it is unspecified whether changes made to HISTSIZE after the history file has been initialized are effective.

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er? ror.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

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>

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion of the listing.

>0 An error occurred.

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

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

Since editors sometimes 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 edi? tor, the command:

```
fc -s | more
```

does not work correctly on many systems.

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

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

EXAMPLES

None.

RATIONALE

This utility is based on the fc built-in of the KornShell.

An early proposal specified the -e option as [-e editor [old= new]], which is not historical practice. Historical practice in fc of either [-e editor] or [-e - [old= new]] is acceptable, but not both to? gether. 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 Some implementations of the KornShell check for the superuser 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 superuser to have history.

HISTSIZE Needed to limit the size of history files. It is the intent of the standard developers 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 diffi? culties of synchronization over a network, the exact nature of the interaction is unspecified.

The initialization process for the history file can be dependent on the system start-up files, in that they may contain commands that effec? tively preempt the settings the user has for HISTFILE and HISTSIZE. For example, function definition commands are recorded in the history file. If the system administrator includes function definitions in some system start-up file called before the ENV file, the history file is initialized before the user can influence its characteristics. In some historical shells, the history file is initialized just after the ENV file has been processed. Because of these situations, the text requires the initialization process to be implementation-defined.

Consideration was given to omitting the fc utility in favor of the com? mand line editing feature in sh. For example, in vi editing mode, typ? ing "<ESC>v" is equivalent to:

```
EDITOR=vi fc
```

However, the fc utility allows the user the flexibility to edit multi? ple 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 - (equiva? lent 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.

Command numbers have no ordering effects; they are like serial numbers.

The -r option and -number operand address the sequence of command exe? cution, regardless of serial numbers. So, for example, if the command number wrapped back to 1 at some arbitrary point, there would be no am? biguity 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 command, regardless of serial number.

FUTURE DIRECTIONS

None.

SEE ALSO

sh

The Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

COPYRIGHT