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 'sh.1p' command

$ man sh.1p

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

sh ? shell, the standard command language interpreter

SYNOPSIS

sh [-abCefhimnuvx] [-o option]... [+abCefhimnuvx] [+o option]...
[command_file [argument...]]

sh -c [-abCefhimnuvx] [-o option]... [+abCefhimnuvx] [+o option]...
command_string [command_name [argument...]]

sh -s [-abCefhimnuvx] [-o option]... [+abCefhimnuvx] [+o option]...
[argument...]

DESCRIPTION

The sh utility is a command language interpreter that shall execute commands read from a command line string, the standard input, or a specified file. The application shall ensure that the commands to be executed are expressed in the language described in Chapter 2, Shell Command Language.

Pathname expansion shall not fail due to the size of a file.

Shell input and output redirections have an implementation-defined off?

set maximum that is established in the open file description.

OPTIONS

The sh utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines, with an exten? sion for support of a leading <plus-sign> ('+') as noted below.

The -a, -b, -C, -e, -f, -m, -n, -o option, -u, -v, and -x options are described as part of the set utility in Section 2.14, Special Built-In Utilities. The option letters derived from the set special built-in shall also be accepted with a leading <plus-sign> ('+') instead of a leading <hyphen-minus> (meaning the reverse case of the option as de? scribed in this volume of POSIX.1?2017).

The following additional options shall be supported:

-c Read commands from the command_string operand. Set the value of special parameter 0 (see Section 2.5.2, Special Parame? ters) from the value of the command_name operand and the po? sitional parameters ($1, $2, and so on) in sequence from the remaining argument operands. No commands shall be read from the standard input.

-i Specify that the shell is interactive; see below. An imple? mentation may treat specifying the -i option as an error if the real user ID of the calling process does not equal the effective user ID or if the real group ID does not equal the effective group ID.

-s Read commands from the standard input.

If there are no operands and the -c option is not specified, the -s op? tion shall be assumed.

If the -i option is present, or if there are no operands and the shell's standard input and standard error are attached to a terminal, the shell is considered to be interactive.

OPERANDS

The following operands shall be supported:

- A single <hyphen-minus> shall be treated as the first operand and then ignored. If both '-' and "--" are given as argu?

ments, or if other operands precede the single <hyphen-mi? nus>, the results are undefined.

argument The positional parameters ($1, $2, and so on) shall be set to arguments, if any.

command_file

The pathname of a file containing commands. If the pathname contains one or more <slash> characters, the implementation attempts to read that file; the file need not be executable.

If the pathname does not contain a <slash> character:

* The implementation shall attempt to read that file from the current working directory; the file need not be exe? cutable.
* If the file is not in the current working directory, the implementation may perform a search for an executable file using the value of PATH, as described in Section 2.9.1.1, Command Search and Execution.

Special parameter 0 (see Section 2.5.2, Special Parameters) shall be set to the value of command_file. If sh is called using a synopsis form that omits command_file, special param? eter 0 shall be set to the value of the first argument passed to sh from its parent (for example, argv[0] for a C program), which is normally a pathname used to execute the sh utility.

command_name

A string assigned to special parameter 0 when executing the commands in command_string. If command_name is not speci? fied, special parameter 0 shall be set to the value of the first argument passed to sh from its parent (for example, argv[0] for a C program), which is normally a pathname used to execute the sh utility.

command_string

A string that shall be interpreted by the shell as one or more commands, as if the string were the argument to the sys? tem() function defined in the System Interfaces volume of

POSIX.1?2017. If the command_string operand is an empty string, sh shall exit with a zero exit status.

STDIN

The standard input shall be used only if one of the following is true:

* The -s option is specified.
* The -c option is not specified and no operands are specified.
* The script executes one or more commands that require input from standard input (such as a read command that does not redirect its input).

See the INPUT FILES section.

When the shell is using standard input and it invokes a command that also uses standard input, the shell shall ensure that the standard in? put file pointer points directly after the command it has read when the command begins execution. It shall not read ahead in such a manner that any characters intended to be read by the invoked command are consumed by the shell (whether interpreted by the shell or not) or that charac? ters that are not read by the invoked command are not seen by the shell. When the command expecting to read standard input is started asynchronously by an interactive shell, it is unspecified whether char? acters are read by the command or interpreted by the shell.

If the standard input to sh is a FIFO or terminal device and is set to non-blocking reads, then sh shall enable blocking reads on standard in? put. This shall remain in effect when the command completes.

INPUT FILES

The input file shall be a text file, except that line lengths shall be unlimited. If the input file consists solely of zero or more blank lines and comments, sh shall exit with a zero exit status.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of sh:

ENV This variable, when and only when an interactive shell is in? voked, shall be subjected to parameter expansion (see Section 2.6.2, Parameter Expansion) by the shell, and the resulting value shall be used as a pathname of a file containing shell

commands to execute in the current environment. The file need not be executable. If the expanded value of ENV is not an absolute pathname, the results are unspecified. ENV shall be ignored if the real and effective user IDs or real and ef? fective group IDs of the process are different.

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. 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.

HOME Determine the pathname of the user's home directory. The con? tents of HOME are used in tilde expansion as described in Section 2.6.1, Tilde Expansion.

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_COLLATE

Determine the behavior of range expressions, equivalence classes, and multi-character collating elements within pat? tern matching.

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), which characters are defined as letters (character class alpha), and the behavior of character classes within pattern matching.

LC_MESSAGES

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

MAIL Determine a pathname of the user's mailbox file for purposes

of incoming mail notification. If this variable is set, the shell shall inform the user if the file named by the variable is created or if its modification time has changed. Informing the user shall be accomplished by writing a string of unspec? ified format to standard error prior to the writing of the next primary prompt string. Such check shall be performed only after the completion of the interval defined by the MAILCHECK variable after the last such check. The user shall be informed only if MAIL is set and MAILPATH is not set.

MAILCHECK

Establish a decimal integer value that specifies how often (in seconds) the shell shall check for the arrival of mail in the files specified by the MAILPATH or MAIL variables. The default value shall be 600 seconds. If set to zero, the shell shall check before issuing each primary prompt.

MAILPATH Provide a list of pathnames and optional messages separated by <colon> characters. If this variable is set, the shell shall inform the user if any of the files named by the vari? able are created or if any of their modification times change. (See the preceding entry for MAIL for descriptions of mail arrival and user informing.) Each pathname can be fol? lowed by '%' and a string that shall be subjected to parame? ter expansion and written to standard error when the modifi? cation time changes. If a '%' character in the pathname is preceded by a <backslash>, it shall be treated as a literal '%' in the pathname. The default message is unspecified.

The MAILPATH environment variable takes precedence over the MAIL variable.

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

PATH Establish a string formatted as described in the Base Defini? tions volume of POSIX.1?2017, Chapter 8, Environment Vari? ables, used to effect command interpretation; see Section

2.9.1.1, Command Search and Execution.

PWD This variable shall represent an absolute pathname of the current working directory. Assignments to this variable may be ignored.

ASYNCHRONOUS EVENTS

The sh utility shall take the standard action for all signals (see Section 1.4, Utility Description Defaults) with the following exceptions.

If the shell is interactive, SIGINT signals received during command line editing shall be handled as described in the EXTENDED DESCRIPTION, and SIGINT signals received at other times shall be caught but no action performed.

If the shell is interactive:

* SIGQUIT and SIGTERM signals shall be ignored.
* If the -m option is in effect, SIGTTIN, SIGTTOU, and SIGTSTP signals shall be ignored.
* If the -m option is not in effect, it is unspecified whether SIGTTIN, SIGTTOU, and SIGTSTP signals are ignored, set to the default action, or caught. If they are caught, the shell shall, in the signal-catching function, set the signal to the default action and raise the signal (after taking any appropriate steps, such as restoring terminal settings).

The standard actions, and the actions described above for interactive shells, can be overridden by use of the trap special built-in utility (see trap and Section 2.11, Signals and Error Handling).

STDOUT

See the STDERR section.

STDERR

Except as otherwise stated (by the descriptions of any invoked utilities or in interactive mode), standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

See Chapter 2, Shell Command Language. The functionality described in the rest of the EXTENDED DESCRIPTION section shall be provided on im? plementations that support the User Portability Utilities option (and the rest of this section is not further shaded for this option).

Command History List

When the sh utility is being used interactively, it shall maintain a list of commands previously entered from the terminal in the file named by the HISTFILE environment variable. The type, size, and internal for? mat of this file are unspecified. Multiple sh processes can share ac? cess to the file for a user, if file access permissions allow this; see the description of the HISTFILE environment variable.

Command Line Editing

When sh is being used interactively from a terminal, the current com? mand and the command history (see fc) can be edited using vi-mode com? mand line editing. This mode uses commands, described below, similar to a subset of those described in the vi utility. Implementations may of? fer other command line editing modes corresponding to other editing utilities.

The command set -o vi shall enable vi-mode editing and place sh into vi insert mode (see Command Line Editing (vi-mode)). This command also shall disable any other editing mode that the implementation may pro? vide. The command set +o vi disables vi-mode editing.

Certain block-mode terminals may be unable to support shell command line editing. If a terminal is unable to provide either edit mode, it need not be possible to set -o vi when using the shell on this termi? nal.

In the following sections, the characters erase, interrupt, kill, and end-of-file are those set by the stty utility.

Command Line Editing (vi-mode)

In vi editing mode, there shall be a distinguished line, the edit line.

All the editing operations which modify a line affect the edit line.

The edit line is always the newest line in the command history buffer.

With vi-mode enabled, sh can be switched between insert mode and com?

mand mode.

When in insert mode, an entered character shall be inserted into the command line, except as noted in vi Line Editing Insert Mode. Upon en? tering sh and after termination of the previous command, sh shall be in insert mode.

Typing an escape character shall switch sh into command mode (see vi Line Editing Command Mode). In command mode, an entered character shall either invoke a defined operation, be used as part of a multi-character operation, or be treated as an error. A character that is not recognized as part of an editing command shall terminate any specific editing command and shall alert the terminal. If sh receives a SIGINT signal in command mode (whether generated by typing the interrupt char? acter or by other means), it shall terminate command line editing on the current command line, reissue the prompt on the next line of the terminal, and reset the command history (see fc) so that the most re? cently executed command is the previous command (that is, the command that was being edited when it was interrupted is not re-entered into the history).

In the following sections, the phrase ``move the cursor to the begin? ning of the word'' shall mean ``move the cursor to the first character of the current word'' and the phrase ``move the cursor to the end of the word'' shall mean ``move the cursor to the last character of the current word''. The phrase ``beginning of the command line'' indicates the point between the end of the prompt string issued by the shell (or the beginning of the terminal line, if there is no prompt string) and the first character of the command text.

vi Line Editing Insert Mode

While in insert mode, any character typed shall be inserted in the cur? rent command line, unless it is from the following set.

<newline> Execute the current command line. If the current command line is not empty, this line shall be entered into the command history (see fc).

erase Delete the character previous to the current cursor position

and move the current cursor position back one character. In insert mode, characters shall be erased from both the screen and the buffer when backspacing.

interrupt If sh receives a SIGINT signal in insert mode (whether gener? ated by typing the interrupt character or by other means), it shall terminate command line editing with the same effects as described for interrupting command mode; see Command Line Editing (vi-mode).

kill Clear all the characters from the input line.

<control>?V

Insert the next character input, even if the character is otherwise a special insert mode character.

<control>?W

Delete the characters from the one preceding the cursor to the preceding word boundary. The word boundary in this case is the closer to the cursor of either the beginning of the line or a character that is in neither the blank nor punct character classification of the current locale.

end-of-file

Interpreted as the end of input in sh. This interpretation shall occur only at the beginning of an input line. If end-of-file is entered other than at the beginning of the line, the results are unspecified.

<ESC> Place sh into command mode.

vi Line Editing Command Mode

In command mode for the command line editing feature, decimal digits not beginning with 0 that precede a command letter shall be remembered. Some commands use these decimal digits as a count number that affects the operation.

The term motion command represents one of the commands:

<space> 0 b F l W ^ $; E f T w | , B e h t

If the current line is not the edit line, any command that modifies the current line shall cause the content of the current line to replace the

content of the edit line, and the current line shall become the edit line. This replacement cannot be undone (see the u and U commands below). The modification requested shall then be performed to the edit line. When the current line is the edit line, the modification shall be done directly to the edit line.

Any command that is preceded by count shall take a count (the numeric value of any preceding decimal digits). Unless otherwise noted, this count shall cause the specified operation to repeat by the number of times specified by the count. Also unless otherwise noted, a count that is out of range is considered an error condition and shall alert the terminal, but neither the cursor position, nor the command line, shall change.

The terms word and bigword are used as defined in the vi description.

The term save buffer corresponds to the term unnamed buffer in vi.

The following commands shall be recognized in command mode:

<newline> Execute the current command line. If the current command line is not empty, this line shall be entered into the command history (see fc).

<control>?L

Redraw the current command line. Position the cursor at the same location on the redrawn line.

Insert the character '#' at the beginning of the current command line and treat the resulting edit line as a comment. This line shall be entered into the command history; see fc.

= Display the possible shell word expansions (see Section 2.6, Word Expansions) of the bigword at the current command line position.

Note: This does not modify the content of the current line, and therefore does not cause the current line to become the edit line.

These expansions shall be displayed on subsequent terminal lines. If the bigword contains none of the characters '?', '*', or '[', an <asterisk> ('*') shall be implicitly assumed

at the end. If any directories are matched, these expansions shall have a '/' character appended. After the expansion, the line shall be redrawn, the cursor repositioned at the current cursor position, and sh shall be placed in command mode.

\ Perform pathname expansion (see Section 2.6.6, Pathname Ex? pansion) on the current bigword, up to the largest set of characters that can be matched uniquely. If the bigword con? tains none of the characters '?', '*', or '[', an <asterisk> ('*') shall be implicitly assumed at the end. This maximal expansion then shall replace the original bigword in the com? mand line, and the cursor shall be placed after this expan? sion. If the resulting bigword completely and uniquely matches a directory, a '/' character shall be inserted di? rectly after the bigword. If some other file is completely matched, a single <space> shall be inserted after the big? word. After this operation, sh shall be placed in insert mode.

* Perform pathname expansion on the current bigword and insert all expansions into the command to replace the current big? word, with each expansion separated by a single <space>. If at the end of the line, the current cursor position shall be moved to the first column position following the expansions and sh shall be placed in insert mode. Otherwise, the current cursor position shall be the last column position of the first character after the expansions and sh shall be placed in insert mode. If the current bigword contains none of the characters '?', '*', or '[', before the operation, an <aster? isk> ('*') shall be implicitly assumed at the end.

@letter Insert the value of the alias named _letter. The symbol let? ter represents a single alphabetic character from the porta? ble character set; implementations may support additional characters as an extension. If the alias _letter contains other editing commands, these commands shall be performed as

part of the insertion. If no alias _letter is enabled, this command shall have no effect.

[count]~ Convert, if the current character is a lowercase letter, to the equivalent uppercase letter and vice versa, as prescribed by the current locale. The current cursor position then shall be advanced by one character. If the cursor was positioned on the last character of the line, the case conversion shall oc? cur, but the cursor shall not advance. If the '~' command is preceded by a count, that number of characters shall be con? verted, and the cursor shall be advanced to the character po? sition after the last character converted. If the count is larger than the number of characters after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[count]. Repeat the most recent non-motion command, even if it was ex? ecuted on an earlier command line. If the previous command was preceded by a count, and no count is given on the '.' command, the count from the previous command shall be in? cluded as part of the repeated command. If the '.' command is preceded by a count, this shall override any count argu? ment to the previous command. The count specified in the '.' command shall become the count for subsequent '.' commands issued without a count.

[number]v Invoke the vi editor to edit the current command line in a temporary file. When the editor exits, the commands in the temporary file shall be executed and placed in the command history. If a number is included, it specifies the command number in the command history to be edited, rather than the current command line.

[count]l (ell)

[count]<space>

Move the current cursor position to the next character posi? tion. If the cursor was positioned on the last character of

the line, the terminal shall be alerted and the cursor shall not be advanced. If the count is larger than the number of characters after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[count]h Move the current cursor position to the countth (default 1) previous character position. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the count is larger than the number of characters before the cursor, this shall not be considered an error; the cursor shall move to the first character on the line.

[count]w Move to the start of the next word. If the cursor was posi? tioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the count is larger than the number of words after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[count]W Move to the start of the next bigword. If the cursor was po? sitioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the count is larger than the number of bigwords after the cursor, this shall not be considered an error; the cursor shall ad? vance to the last character on the line.

[count]e Move to the end of the current word. If at the end of a word, move to the end of the next word. If the cursor was posi? tioned on the last character of the line, the terminal shall be alerted and the cursor shall not be advanced. If the count is larger than the number of words after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[count]E Move to the end of the current bigword. If at the end of a bigword, move to the end of the next bigword. If the cursor

was positioned on the last character of the line, the termi? nal shall be alerted and the cursor shall not be advanced. If the count is larger than the number of bigwords after the cursor, this shall not be considered an error; the cursor shall advance to the last character on the line.

[count]b Move to the beginning of the current word. If at the begin? ning of a word, move to the beginning of the previous word. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the count is larger than the number of words preceding the cursor, this shall not be considered an error; the cursor shall return to the first character on the line.

[count]B Move to the beginning of the current bigword. If at the be? ginning of a bigword, move to the beginning of the previous bigword. If the cursor was positioned on the first character of the line, the terminal shall be alerted and the cursor shall not be moved. If the count is larger than the number of bigwords preceding the cursor, this shall not be considered an error; the cursor shall return to the first character on the line.

^ Move the current cursor position to the first character on the input line that is not a <blank>.

$ Move to the last character position on the current command line.

0 (Zero.) Move to the first character position on the current command line.

[count]| Move to the countth character position on the current command line. If no number is specified, move to the first position. The first character position shall be numbered 1. If the count is larger than the number of characters on the line, this shall not be considered an error; the cursor shall be placed on the last character on the line.

[count]fc Move to the first occurrence of the character 'c' that occurs

after the current cursor position. If the cursor was posi?
tioned on the last character of the line, the terminal shall
be alerted and the cursor shall not be advanced. If the char?
acter 'c' does not occur in the line after the current cursor
position, the terminal shall be alerted and the cursor shall
not be moved.

[count]Fc Move to the first occurrence of the character 'c' that occurs
before the current cursor position. If the cursor was posi?
tioned on the first character of the line, the terminal shall
be alerted and the cursor shall not be moved. If the charac?
ter 'c' does not occur in the line before the current cursor
position, the terminal shall be alerted and the cursor shall
not be moved.

[count]tc Move to the character before the first occurrence of the
character 'c' that occurs after the current cursor position.
If the cursor was positioned on the last character of the
line, the terminal shall be alerted and the cursor shall not
be advanced. If the character 'c' does not occur in the line
after the current cursor position, the terminal shall be
alerted and the cursor shall not be moved.

[count]Tc Move to the character after the first occurrence of the char?
acter 'c' that occurs before the current cursor position. If
the cursor was positioned on the first character of the line,
the terminal shall be alerted and the cursor shall not be
moved. If the character 'c' does not occur in the line before
the current cursor position, the terminal shall be alerted
and the cursor shall not be moved.

[count]; Repeat the most recent f, F, t, or T command. Any number ar?
gument on that previous command shall be ignored. Errors are
those described for the repeated command.

[count], Repeat the most recent f, F, t, or T command. Any number ar?
gument on that previous command shall be ignored. However,
reverse the direction of that command.

a Enter insert mode after the current cursor position. Charac?
ters that are entered shall be inserted before the next char?
acter.

A Enter insert mode after the end of the current command line.

i Enter insert mode at the current cursor position. Characters
that are entered shall be inserted before the current charac?
ter.

I Enter insert mode at the beginning of the current command
line.

R Enter insert mode, replacing characters from the command line
beginning at the current cursor position.

[count]cmotion
Delete the characters between the current cursor position and
the cursor position that would result from the specified mo?
tion command. Then enter insert mode before the first charac?
ter following any deleted characters. If count is specified,
it shall be applied to the motion command. A count shall be
ignored for the following motion commands:

0 ^ $ c

If the motion command is the character 'c', the current com?
mand line shall be cleared and insert mode shall be entered.
If the motion command would move the current cursor position
toward the beginning of the command line, the character under
the current cursor position shall not be deleted. If the mo?
tion command would move the current cursor position toward
the end of the command line, the character under the current
cursor position shall be deleted. If the count is larger
than the number of characters between the current cursor po?
sition and the end of the command line toward which the mo?
tion command would move the cursor, this shall not be consid?
ered an error; all of the remaining characters in the afore?
mentioned range shall be deleted and insert mode shall be en?
tered. If the motion command is invalid, the terminal shall

be alerted, the cursor shall not be moved, and no text shall be deleted.

C Delete from the current character to the end of the line and enter insert mode at the new end-of-line.

S Clear the entire edit line and enter insert mode.

[count]rc Replace the current character with the character 'c'. With a number count, replace the current and the following count-1 characters. After this command, the current cursor position shall be on the last character that was changed. If the count is larger than the number of characters after the cursor, this shall not be considered an error; all of the remaining characters shall be changed.

[count]_ Append a <space> after the current character position and then append the last bigword in the previous input line after the <space>. Then enter insert mode after the last character just appended. With a number count, append the countth big? word in the previous line.

[count]x Delete the character at the current cursor position and place the deleted characters in the save buffer. If the cursor was positioned on the last character of the line, the character shall be deleted and the cursor position shall be moved to the previous character (the new last character). If the count is larger than the number of characters after the cursor, this shall not be considered an error; all the characters from the cursor to the end of the line shall be deleted.

[count]X Delete the character before the current cursor position and place the deleted characters in the save buffer. The charac? ter under the current cursor position shall not change. If the cursor was positioned on the first character of the line, the terminal shall be alerted, and the X command shall have no effect. If the line contained a single character, the X command shall have no effect. If the line contained no char? acters, the terminal shall be alerted and the cursor shall

not be moved. If the count is larger than the number of char?
acters before the cursor, this shall not be considered an er?
ror; all the characters from before the cursor to the begin?
ning of the line shall be deleted.

[count]dmotion

Delete the characters between the current cursor position and the character position that would result from the motion com?
mand. A number count repeats the motion command count times. If the motion command would move toward the beginning of the command line, the character under the current cursor position shall not be deleted. If the motion command is d, the entire current command line shall be cleared. If the count is larger than the number of characters between the current cursor po?
sition and the end of the command line toward which the mo?
tion command would move the cursor, this shall not be consid?
ered an error; all of the remaining characters in the afore?
mentioned range shall be deleted. The deleted characters shall be placed in the save buffer.

D Delete all characters from the current cursor position to the end of the line. The deleted characters shall be placed in the save buffer.

[count]ymotion

Yank (that is, copy) the characters from the current cursor position to the position resulting from the motion command into the save buffer. A number count shall be applied to the motion command. If the motion command would move toward the beginning of the command line, the character under the cur?
rent cursor position shall not be included in the set of yanked characters. If the motion command is y, the entire current command line shall be yanked into the save buffer. The current cursor position shall be unchanged. If the count is larger than the number of characters between the current cursor position and the end of the command line toward which

the motion command would move the cursor, this shall not be considered an error; all of the remaining characters in the aforementioned range shall be yanked.

Y Yank the characters from the current cursor position to the end of the line into the save buffer. The current character position shall be unchanged.

[count]p Put a copy of the current contents of the save buffer after the current cursor position. The current cursor position shall be advanced to the last character put from the save buffer. A count shall indicate how many copies of the save buffer shall be put.

[count]P Put a copy of the current contents of the save buffer before the current cursor position. The current cursor position shall be moved to the last character put from the save buf? fer. A count shall indicate how many copies of the save buf? fer shall be put.

u Undo the last command that changed the edit line. This opera? tion shall not undo the copy of any command line to the edit line.

U Undo all changes made to the edit line. This operation shall not undo the copy of any command line to the edit line.

[count]k

[count]- Set the current command line to be the countth previous com? mand line in the shell command history. If count is not spec? ified, it shall default to 1. The cursor shall be positioned on the first character of the new command. If a k or - com? mand would retreat past the maximum number of commands in ef? fect for this shell (affected by the HISTSIZE environment variable), the terminal shall be alerted, and the command shall have no effect.

[count]j

[count]+ Set the current command line to be the countth next command line in the shell command history. If count is not specified,

it shall default to 1. The cursor shall be positioned on the first character of the new command. If a j or + command ad? vances past the edit line, the current command line shall be restored to the edit line and the terminal shall be alerted.

[number]G Set the current command line to be the oldest command line stored in the shell command history. With a number number, set the current command line to be the command line number in the history. If command line number does not exist, the ter? minal shall be alerted and the command line shall not be changed.

/pattern<newline>

Move backwards through the command history, searching for the specified pattern, beginning with the previous command line. Patterns use the pattern matching notation described in Sec? tion 2.13, Pattern Matching Notation, except that the '^' character shall have special meaning when it appears as the first character of pattern. In this case, the '^' is dis? carded and the characters after the '^' shall be matched only at the beginning of a line. Commands in the command history shall be treated as strings, not as filenames. If the pattern is not found, the current command line shall be unchanged and the terminal shall be alerted. If it is found in a previous line, the current command line shall be set to that line and the cursor shall be set to the first character of the new command line.

If pattern is empty, the last non-empty pattern provided to / or ? shall be used. If there is no previous non-empty pat? tern, the terminal shall be alerted and the current command line shall remain unchanged.

?pattern<newline>

Move forwards through the command history, searching for the specified pattern, beginning with the next command line. Pat? terns use the pattern matching notation described in Section

2.13, Pattern Matching Notation, except that the '^' charac?
ter shall have special meaning when it appears as the first
character of pattern. In this case, the '^' is discarded and
the characters after the '^' shall be matched only at the be?
ginning of a line. Commands in the command history shall be
treated as strings, not as filenames. If the pattern is not
found, the current command line shall be unchanged and the
terminal shall be alerted. If it is found in a following
line, the current command line shall be set to that line and
the cursor shall be set to the fist character of the new com?
mand line.

If pattern is empty, the last non-empty pattern provided to /
or ? shall be used. If there is no previous non-empty pat?
tern, the terminal shall be alerted and the current command
line shall remain unchanged.

n Repeat the most recent / or ? command. If there is no previ?
ous / or ?, the terminal shall be alerted and the current
command line shall remain unchanged.

N Repeat the most recent / or ? command, reversing the direc?
tion of the search. If there is no previous / or ?, the ter?
minal shall be alerted and the current command line shall re?
main unchanged.

EXIT STATUS

The following exit values shall be returned:

0 The script to be executed consisted solely of zero or more
blank lines or comments, or both.

1?125 A non-interactive shell detected an error other than com?
mand_file not found or executable, including but not limited to
syntax, redirection, or variable assignment errors.

126 A specified command_file could not be executed due to an
[ENOEXEC] error (see Section 2.9.1.1, Command Search and Execu?
tion, item 2).

127 A specified command_file could not be found by a non-interac?

tive shell.

Otherwise, the shell shall return the exit status of the last command it invoked or attempted to invoke (see also the exit utility in Section 2.14, Special Built-In Utilities).

CONSEQUENCES OF ERRORS

See Section 2.8.1, Consequences of Shell Errors.

The following sections are informative.

APPLICATION USAGE

Standard input and standard error are the files that determine whether a shell is interactive when -i is not specified. For example:

```
sh > file
```

and:

```
sh 2> file
```

create interactive and non-interactive shells, respectively. Although both accept terminal input, the results of error conditions are differ? ent, as described in Section 2.8.1, Consequences of Shell Errors; in the second example a redirection error encountered by a special built- in utility aborts the shell.

A conforming application must protect its first operand, if it starts with a <plus-sign>, by preceding it with the "--" argument that denotes the end of the options.

Applications should note that the standard PATH to the shell cannot be assumed to be either /bin/sh or /usr/bin/sh, and should be determined by interrogation of the PATH returned by getconf PATH, ensuring that the returned pathname is an absolute pathname and not a shell built-in. For example, to determine the location of the standard sh utility:

```
command -v sh
```

On some implementations this might return:

```
/usr/xpg4/bin/sh
```

Furthermore, on systems that support executable scripts (the "#!" con? struct), it is recommended that applications using executable scripts install them using getconf PATH to determine the shell pathname and up? date the "#!" script appropriately as it is being installed (for exam?

ple, with sed). For example:

```
#
# Installation time script to install correct POSIX shell pathname
#
# Get list of paths to check
#
Sifs=$IFS
Sifs_set=${IFS+y}
IFS=:
set -- $(getconf PATH)
if [ "$Sifs_set" = y ]
then
    IFS=$Sifs
else
    unset IFS
fi
#
# Check each path for 'sh'
#
for i
do
    if [ -x "${i}"/sh ]
    then
        Pshell=${i}/sh
    fi
done
#
# This is the list of scripts to update. They should be of the
# form '${name}.source' and will be transformed to '${name}'.
# Each script should begin:
#
# #!INSTALLSHELLPATH
#
```

```
scripts="a b c"
#
# Transform each script
#
for i in ${scripts}
do
    sed -e "s|INSTALLSHELLPATH|${Pshell}|" < ${i}.source > ${i}
done
```

EXAMPLES

1. Execute a shell command from a string:

   ```
   sh -c "cat myfile"
   ```

2. Execute a shell script from a file in the current directory:

   ```
   sh my_shell_cmds
   ```

RATIONALE

The sh utility and the set special built-in utility share a common set of options.

The name IFS was originally an abbreviation of ``Input Field Separa? tors''; however, this name is misleading as the IFS characters are ac? tually used as field terminators. One justification for ignoring the contents of IFS upon entry to the script, beyond security considera? tions, is to assist possible future shell compilers. Allowing IFS to be imported from the environment prevents many optimizations that might otherwise be performed via dataflow analysis of the script itself.

The text in the STDIN section about non-blocking reads concerns an in? stance of sh that has been invoked, probably by a C-language program, with standard input that has been opened using the O_NONBLOCK flag; see open() in the System Interfaces volume of POSIX.1?2017. If the shell did not reset this flag, it would immediately terminate because no in? put data would be available yet and that would be considered the same as end-of-file.

The options associated with a restricted shell (command name rsh and the -r option) were excluded because the standard developers considered that the implied level of security could not be achieved and they did

not want to raise false expectations.

On systems that support set-user-ID scripts, a historical trapdoor has been to link a script to the name -i. When it is called by a sequence such as:

```
sh -
```

or by:

```
#! usr/bin/sh -
```

the historical systems have assumed that no option letters follow. Thus, this volume of POSIX.1?2017 allows the single <hyphen-minus> to mark the end of the options, in addition to the use of the regular "--" argument, because it was considered that the older practice was so per? vasive. An alternative approach is taken by the KornShell, where real and effective user/group IDs must match for an interactive shell; this behavior is specifically allowed by this volume of POSIX.1?2017.

Note: There are other problems with set-user-ID scripts that the two approaches described here do not resolve.

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 user's settings of HISTFILE and HISTSIZE. For exam? ple, function definition commands are recorded in the history file, un? less the set -o nolog option is set. If the system administrator in? cludes function definitions in some system start-up file called before the ENV file, the history file is initialized before the user gets a chance to influence its characteristics. In some historical shells, the history file is initialized just after the ENV file has been processed. Therefore, it is implementation-defined whether changes made to HIST? FILE after the history file has been initialized are effective.

The default messages for the various MAIL-related messages are unspeci? fied because they vary across implementations. Typical messages are:

```
"you have mail\n"
```

or:

```
"you have new mail\n"
```

It is important that the descriptions of command line editing refer to

the same shell as that in POSIX.1?2008 so that interactive users can also be application programmers without having to deal with program? matic differences in their two environments. It is also essential that the utility name sh be specified because this explicit utility name is too firmly rooted in historical practice of application programs for it to change.

Consideration was given to mandating a diagnostic message when attempt? ing to set vi-mode on terminals that do not support command line edit? ing. However, it is not historical practice for the shell to be cog? nizant of all terminal types and thus be able to detect inappropriate terminals in all cases. Implementations are encouraged to supply diag? nostics in this case whenever possible, rather than leaving the user in a state where editing commands work incorrectly.

In early proposals, the KornShell-derived emacs mode of command line editing was included, even though the emacs editor itself was not. The community of emacs proponents was adamant that the full emacs editor not be standardized because they were concerned that an attempt to standardize this very powerful environment would encourage vendors to ship strictly conforming versions lacking the extensibility required by the community. The author of the original emacs program also expressed his desire to omit the program. Furthermore, there were a number of historical systems that did not include emacs, or included it without supporting it, but there were very few that did not include and support vi. The shell emacs command line editing mode was finally omitted be? cause it became apparent that the KornShell version and the editor be? ing distributed with the GNU system had diverged in some respects. The author of emacs requested that the POSIX emacs mode either be deleted or have a significant number of unspecified conditions. Although the KornShell author agreed to consider changes to bring the shell into alignment, the standard developers decided to defer specification at that time. At the time, it was assumed that convergence on an accept? able definition would occur for a subsequent draft, but that has not happened, and there appears to be no impetus to do so. In any case, im?

plementations are free to offer additional command line editing modes based on the exact models of editors their users are most comfortable with.

Early proposals had the following list entry in vi Line Editing Insert Mode:

\ If followed by the erase or kill character, that character shall be inserted into the input line. Otherwise, the <backslash> it? self shall be inserted into the input line.

However, this is not actually a feature of sh command line editing in? sert mode, but one of some historical terminal line drivers. Some con? forming implementations continue to do this when the stty iexten flag is set.

In interactive shells, SIGTERM is ignored so that kill 0 does not kill the shell, and SIGINT is caught so that wait is interruptible. If the shell does not ignore SIGTTIN, SIGTTOU, and SIGTSTP signals when it is interactive and the -m option is not in effect, these signals suspend the shell if it is not a session leader. If it is a session leader, the signals are discarded if they would stop the process, as required by the System Interfaces volume of POSIX.1?2017, Section 2.4.3, Signal Ac? tions for orphaned process groups.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.9.1.1, Command Search and Execution, Chapter 2, Shell Command Language, cd, echo, exit, fc, pwd, invalid, set, stty, test, trap, umask, vi

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

The System Interfaces volume of POSIX.1?2017, dup(), exec, exit(), fork(), open(), pipe(), signal(), system(), ulimit(), umask(), wait()

COPYRIGHT

table Operating System Interface (POSIX), The Open Group Base Specifi? cations Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker? nel.org/doc/man-pages/reporting_bugs.html .

IEEE/The Open Group 2017 SH(1P)