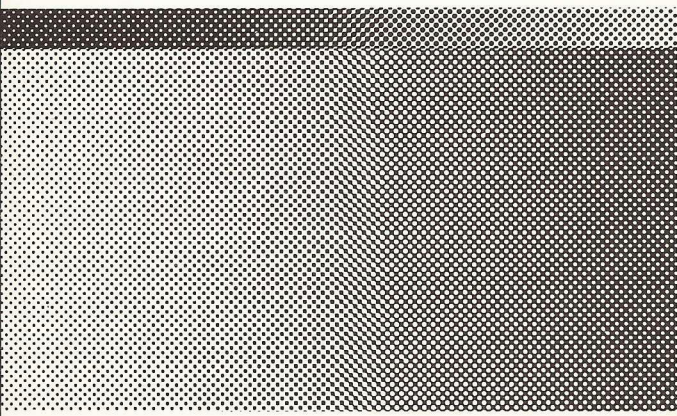
UNIX® System V
Visual Editor
Quick Reference

307-262
Issue 1

UNIX® System V - Release 2
Visual Editor
Quick Reference



NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

TRADEMARKS

The following trademarks are used in this manual:

- UNIX is a registered trademark of AT&T.

- The visual editor is based on software developed by The University of California, Berkeley, California; Computer Science Division, Department of Electrical Engineering and Computer Science, and such software is owned and licensed by the Regents of the University of California.

ORDERING INFORMATION

Additional copies of this document can be ordered by calling

1-800-432-6600 Inside the U.S.A.

OR

1-317-352-8557 Outside the U.S.A.

OR by writing to:

AT&T Customer Information Center
Attn: Customer Service Representative
P.O. Box 19901
Indianapolis, IN 46219

VI EDITOR QUICK REFERENCE

PAGE

VI EDITOR QUICK REFERENCE

INTRODUCTION

This quick reference is based on UNIX System V. To use the full screen editing capabilities of the visual (vi) editor, your terminal must be defined by the system. You must then tell the system about your terminal as shown in **Specifying Terminal Type**. If your terminal is not defined by the system, you can use the vi editor in the open mode. Most vi commands will work in the open mode, but full screen editing is not provided.

Specifying Terminal Type

TERM=*type*	*Type* defined in system **terminfo** file.
export TERM	Allows **vi** to be used in any subshell.

Notation

^	Denotes the CONTROL key to be held down while the following character is typed.
↑	Used to show the caret (^) should be typed.
[n]	Optional number of repetitions preceding a command. Do not type [ ]. In most cases omitting *n* defaults to one.
object	The text object—(character, word, sentence, paragraph, or line) that a command operates on.

Special Keys

:	A prefix to a set of commands for file and option manipulation and escapes to the shell. The : and later keystrokes appear at the bottom of the screen. The command is terminated with a **<CR>** or **<ESC>**.
<ESC>	ESCAPE key used to return to command mode. Type **<ESC>** when you are not sure of the current mode. Causes a beep if already in command mode (harmless).
<CR>	Carriage RETURN key.
BS	BACKSPACE key. **^H** on terminals without a backspace key.
DELETE	Sometimes labeled **DEL**, **BREAK**, or **RUBOUT**. This key generates an interrupt that tells the editor to stop what it is doing.

ENTERING VI EDITOR

Note: Follow entry with <CR>

vi *file*	Edit at first line of *file*
vi	Edit a new empty *file*
vi + *n file*	Edit at *n* line in *file*
vi + *file*	Edit at last line in *file*
vi -r	List saved files
vi -r *file*	Recover *file* and edit saved *file*
vi *file1, file2* ...	Edit *file1*; *file2*; ... (After *file1* enter **:n** for each remaining file.)
vi -t *tag*	Edit at *tag* file in *tags* file
vi +/*pat file*	Search for and edit at *pattern* in file
view *file*	Read-only view of file

LEAVING VI EDITOR

:q<CR>	Quit vi when no changes have occurred since last write
:q!<CR>	Quit vi; do not save changes since last write
:wq<CR>	Write and quit (exit vi, saving changes)
ZZ	Write and quit (exit vi, saving changes)

POSITIONING THE CURSOR

File Positioning

[n]^**f**	Forward *[n]* full screens
[n]^**b**	Backward screens
[n]^**d**	Scroll down (default is half screen)
[n]^**u**	Scroll up (default is half screen)
[n]^**e**	Scroll down 1 line
[n]^**y**	Scroll up 1 line
*[n]***G**	Goto line *n* (default is last line of file)
[n]/pat	Next line matching *pat*
[n]?*pat*	Previous line matching *pat*
*[n]***n**	Repeat last / or ?
*[n]***N**	Reverse last / or ?
[n]/pat/+m	*mth* line after *pat*
[n]?*pat*?*-m*	*mth* line before *pat*

Screen Positioning

*[n]***H**	To *nth* line from top of display. Without *n* to top
*[n]***L**	To *nth* line from bottom of display. Without *n* to bottom
M	To middle line of display

Line Positioning

0	Beginning of line
[n]$	End of line
[n]+	Next line, at first non-white
[n]−	Previous line, at first non-white
[n]<CR>	Return, same as +
[n]↓ or j	Next line, same column
[n]↑ or k	Previous line, same column

Character Positioning Within a Line

[n]↑	First non-white
*[n]***l** or →	Forward one character
*[n]***h** or ←	Backward one character
*[n]*spacebar	Same as →
*[n]*backspace	Backward one character
[n]^**h**	Same as ← or backspace
*[n]***f***x*	Find *x* forward
*[n]***F***x*	Find *x* backward
*[n]***t***x*	Move up to *x* forward
*[n]***T***x*	Move up to *x* backward
[n];	Repeat last **f**, **F**, **t**, or **T**
[n],	Inverse of ;
[n]|	Move to specified column number *[n]*

Word Positioning

*[n]***w**	Move forward to beginning of word. Punctuation and strings of punctuation count as words.
*[n]***b**	Move back to beginning of word. Punctuation and strings of punctuation count as words.
*[n]***e**	Move forward to end of word. Punctuation and strings of punctuation count as words.
*[n]***W**	Move forward to beginning of word. Punctuation ignored.
*[n]***B**	Move back to beginning of word. Punctuation ignored.
*[n]***E**	Move forward to end of word. Punctuation ignored.

Sentence, Paragraph, Heading Positioning

[n])	Forward to next sentence
[n](	Back a sentence
[n]}	Forward to next paragraph
[n]{	Back a paragraph
[n]]]	Forward to next heading
[n][[	Back a heading

CREATING TEXT

a*text*<ESC>	Append after cursor, until <ESC>
i*text*<ESC>	Insert before cursor
A*text*<ESC>	Append at end of line
I*text*<ESC>	Insert before first non-blank
o*text*<ESC>	Open line below
O*text*<ESC>	Open above

MAKING CORRECTIONS DURING TEXT CREATION

^**w**	Erase last word during an insert
kill	Kill the insert on this line (usually @)
backspace	Erase last character
^**h**	Erase last character
<ESC>	Ends insertion, back to command mode
^**v**	To print non-printing character, precede non-printing character with ^v

MODIFYING TEXT

Changing Text

~	Switch character from lowercase to uppercase and vise versa
*[n]***C***text*<ESC>	Change from cursor to end of line (same as c$)

*[n]***R***text*<ESC>	Replace characters
*[n]***S***text*<ESC>	Substitute on lines
*[n]***c**obj*text*<ESC>	Change the specified object (word) to the following *text*
*[n]***r***x*	Replace character with *x*
*[n]***s***text*<ESC>	Replace a character with a *text* string
*[n]***cc***text*<ESC>	Change a whole line

Deleting Text

D	Delete from cursor to end of line
*[n]***x**	Delete a character
*[n]***X**	Delete character to left of cursor
*[n]***d***(object)*	Delete the specified *object* (word, sentence, paragraph, etc.)
*[n]***dd**	Delete a line

Moving Text

" *r*	Named register *r* that saves delete commands. Legal values of *r* are letters **a** through **z**.
" *r***p**	Puts deleted text from register *r* after or below cursor
" *r***P**	Puts deleted text from register *r* before or above cursor
p	Puts last deleted text after or below cursor

P	Puts last deleted text before or above cursor

Copying Text

" *r*	Named register *r* that can precede a yank command. Legal values of *r* are letters **a** through **z**, e.g.: " **a2yy** places the current line and the next line in buffer **a**.
" *R*	Named register *R* that can precede a yank command. Legal values of *R* are letters **A** through **Z**. This command appends yanked information into associated buffer *r* above, e.g.: " **Ay) appends the following sentence to register a.**
y*[n]*object	Yanks a copy of the following object into a register
*[n]***Y**	Yanks a copy of the current line into a register
*[n]***yy**	Same as **Y**
" *r***p**	Puts yanked text from register *r* after or below cursor
" *r***P**	Puts yanked text from register *r* before or above cursor
p	Puts last yanked text after or below cursor
P	Puts last yanked text before or above cursor

UNDOING, REDOING, RETRIEVING

u	Undo last change
U	Restore current line
.	Repeat last change
" *h***p**	Retrieve one of last 9 deletes; *h* is a hidden register numbered 1 through 9. Retrieved in reverse order.

DOING GLOBAL SEARCHES AND CHANGES

Note: Follow entry with <CR>

:g/*text*	Move cursor to last line in file with *text*
:g/*text***/p**	Print all lines with *text*
:g/*text***/nu**	Print all lines and line numbers with *text*
:*[m],[n]***g/***text*	Move cursor to *n* line in file with *text*
:*[m],[n]***g/***text***/p**	Print all lines with *text* from line *m* to *n*
:*[m],[n]***g/***text***/nu**	Print all lines and line numbers with *text* from line *m* to *n*
:g/*text***/s//***newtext*	Change first appearance of *text* in each line in file to *newtext*

:g/*text*/**s**//*newtext*/**p**	Change first appearance of *text* in each line in file to *newtext* and print each changed line
:g/*text*/**s**//*newtext*/**c**	List one at a time each line with *text* and change as required to *newtext* using a **y**<CR>
:*[m],[n]***g**/*text*/**s**//*newtext*	Change first appearance of *text* in each line in file to *newtext*
:*[m],[n]***g**/*text*/**s**//*newtext*/**p**	Change first appearance of *text* in lines from *m* to *n* to *newtext* and print each changed line
:*[m],[n]***g**/*text*/**s**//*newtext*/**c**	List one at a time each line with *text* from *m* to *n* and change as required to *newtext* using a **y**<CR>

MANIPULATING FILES

Copy From Another File

:r *file*<CR>	Copy *file* into buffer after current line
:*[n]***r** *file*<CR>	Copy *file* to buffer after *nth* line

Copy To Another File

Note: Follow entry with <CR>

:w *file*	Write the current file to *file*
:w! *file*	Overwrite existing *file* with *file*
:w>>*file*	Add current file to end of *file*
:*[m],[n]***w** *file*	Write lines *m* through *n* to *file*
:*[m],[n]***w!** *file*	Overwrite existing *file* with *file* containing lines *m* through *n*
:*[m],[n]***w>>***file*	Add lines *m* through *n* to end of *file*

Edit Current File

:w<CR>	Write changes to current file
:w *file*<CR>	Write *file* to current unnamed file
:e!<CR>	Reedit current file, discarding changes since last write
:f<CR>	Show current file and line
^G	Synonym for **:f**
:ta *tag*<CR>	To tag file entry *tag*
^]	**:ta**, following word is *tag*

Edit Other Files From Current File

:e *file*<CR>	Edit *file* when write has occurred in current file; return to shell after edit. Changes not lost in current file
:e! *file*<CR>	Edit *file* when no write has occurred in current file; return to shell after edit. Changes lost in current file
:e + *name*<CR>	Edit starting at end
:e + *n*<CR>	Edit starting at line *n*
:n<CR>	Edit next file in list when vi was called with more than one file
:n *args*<CR>	Specify new list of files to be edited
:e #<CR>	Edit alternate file when two files are being edited
^↑	Synonym for **:e #**

ESCAPING TO THE SHELL

:sh<CR>	Start a separate shell (to run several commands), return with ^D
:!*command*<CR>	Run one shell *command* , then return to current buffer

MARKING AND RETURNING

`	Previous context
'	... at first non-white in line
m*x*	Mark position with letter *x*
`*x*	to mark *x*
'*x*	... at first non-white in line

MISCELLANEOUS OPERATIONS

.	Repeat the last append, insert, open, delete, change, or put command
~	Switch character from lowercase to uppercase and vice versa
^?	Delete or rub out interrupts
i<CR><ESC>	Split a line before the cursor
a<CR><ESC>	Split a line after the cursor
^L	Reprint screen if ^? scrambles it
J	Join lines
:nu<CR>	Line number cursor is on
xp	Switch characters

SETTING OPTIONS

Initializing Options

:set *x*<CR>	Enable option *x*
:set no*x*<CR>	Disable option *x*
:set *x*=*val*<CR>	Assign a value to *x* option
:set<CR>	Show changed options
:set all<CR>	Show all options
:set *x*?<CR>	Show value of option *x*

Options

autoindent, ai (default: noai)

When on, in the append, change, insert, open, or substitute mode a new line will be started at same indent as previous line.

autoprint,ap (default: ap)

Causes the current line to be printed after each delete, copy, join, move, substitute, **t**, undo or shift command. This has the same effect as supplying a trailing **p** to each such command. The *autoprint* is suppressed in globals and only applies to the last of many commands on a line.

autowrite,aw (default: noaw)

Causes the contents of the buffer to be written to the current file (if you have modified it) and gives a next, rewind, tab, or ! command, or a ^↑ (switch

files) or ^] (tag goto) command. **Note**: The command does not autowrite. In each case, there is an equivalent way of switching when the *autowrite* option is set to avoid the autowrite (ex for next, rewind! for rewind, tag! for tag, shell for !, and :e #nd a :ta! command.

beautify, bf (default: nobeautify)

Causes all control characters except tab, new-line, and form-feed to be discarded from the input. A complaint is registered the first time a backspace character is discarded. The *beautify* option does not apply to command input.

directory, dir (default: dir=/tmp)

Specifies the directory in which **vi** places its buffer file. If this directory in not writable, then the editor will exit abruptly when it fails to be able to create its buffer there.

edcompatible (default: noedcompatible)

Causes the presence or absence of **g** and **c** suffixes on substitute commands to be remembered and to be toggled by repeating the suffices. The suffix **r** makes the substitution be as in the ~ command, instead of like **&**.

errorbells,eb (default: noeb)

Error messages are preceded by a bell. Bell ringing in *open* and *visual* mode on errors is not suppressed by setting *noeb*. If possible the editor always places the error message in a standout mode of the

terminal (such as inverse video) instead of ringing the bell.

hardtabs, ht (default: ht=8)

Gives the boundaries on which terminal hardware tabs are set (or on which the system expands tabs).

ignorecase,ic (default: noic)

All uppercase characters in the text are mapped to lowercase in regular expression matching and vice versa, except in character class specifications.

lisp (default: nolisp)

The *autoindent* option indents appropriately for *lisp* code, and the **()**, {}, [[, and]] commands in *open* and *visual* modes are modified to have meaning for *lisp.*

list (default: nolist)

All printed lines will be displayed more unambiguously, showing tabs and end-of-lines as in the *list* command.

magic (default: magic for **vi**)

If *nomagic* is set, the number of regular expression metacharacters is greatly reduced, with only ↑ and $ having special effects. In addition, the metacharacters ~ and **&** of the replacement pattern are treated as normal characters. All the normal metacharacters may be made *magic* when *nomagic* is set by preceding them with a \.

mesg (default: mesg)

Causes write permission to be turned off to the terminal while you are in *visual* mode if *nomesg* is set.

number,nu (default: nonumber)

Causes all output lines to be printed with line numbers. In addition, each input line will be prompted for by supplying the line number it will have.

open (default: open)

If *noopen*, the commands *open* and *visual* are not permitted.

optimize, opt (default: optimize)

Throughput of text is expedited by setting the terminal not to do automatic carriage returns when printing more than one (logical) line of output, greatly speeding output on terminals without addressable cursors when text with leading white space is printed.

paragraphs, para (default: para=IPLPPPQPP LIbp)

Specifies the paragraphs for the { and } operations in *open* and *visual* mode. The pairs of characters in the option's value are the names of the macros which start paragraphs.

prompt (default: prompt)

Command mode input is prompted for with a colon (:).

readonly (default:noreadonly)

Set by chmod shell command to allow read but no write.

redraw (default: noredraw)

The editor simulates (using great amounts of output) an intelligent terminal on a dumb terminal (e.g., during insertions in *visual* the characters to the right of the cursor position are refreshed as each input character is typed). This option is useful only at very high speed.

remap (default: remap)

If on, macros are repeatedly tried until they are unchanged. For example, if **o** is mapped to **O**, and **O** is mapped to **I**, then if *remap* is set, **o** will map to **I**; but if *noremap* is set, it will map to **O**.

report (default: report=5)

Specifies a threshold for feedback from commands. Any command which modifies more than the specified number of lines will provide feedback as to the scope of its changes. For commands such as global, *open*, undo, and *visual*, which have potentially more far-reaching scope, the net change in the number of lines in the buffer is presented at the end of the command subject to this same threshold. Thus, notification is suppressed during a global command on the individual commands performed.

scroll (default: scroll=½ window)

Determines the number of logical lines scrolled when an end-of-file is received from a terminal input in *command* mode and the number of lines printed by a *command* mode **z** command (double the value of *scroll*).

sections (default: sections=SHNHH HU)

Specifies the section macros for the [[and]] operations in *open* and *visual* modes. The pairs of characters in the option's value are the names of the macros which start paragraphs.

shell, sh (default: sh=/bin/sh)

Gives the path name of the shell forked for the shell escape command **!**, and by the *shell* command. The default is taken from SHELL in the environment, if present.

shiftwidth, sw (default: sw=8)

Gives the width a software tabstop used in reverse tabbing with **^D** when using *autoindent* to append text, and by the shift commands.

showmatch, sm (default: nosm)

In *open* and *visual* mode, when a) or } is typed, the cursor moves to the matching (or { for one second if this matching character is on the screen. Extremely useful with *lisp*.

slowopen, slow (terminal-dependent)

Affects the display algorithm used in *visual* mode, holding off display updating during input of new text to improve throughput when the terminal in use is both slow and unintelligent.

tabstop, ts (default: ts=8)

The editor expands tabs in the input file to be on *tabstop* boundaries for the purposes of display.

taglength, tl (default: tl=0)

Tags are not significant beyond this many characters. A value of zero (the default) means that all characters are significant.

tags (default: tags=tags/usr/lib/tags)

A path of files to be used as tag files for the *tag* command. A requested tag is searched for in the specified files, sequentially. By default, files called *tags* are searched for in the current directory and in */usr/lib* (a master file for the entire system).

term (from environment $TERM)

The terminal type of the output device.

terse (default: noterse)

Shorter error diagnostics are produced for the experienced user.

ttytype=

Terminal type defined to system for visual mode. Can be defined before entering visual editor by TERM=type.

warn (default: warn)

Warns if there has been "[No write since last change]" before a ! command escape.

window (default: window=speed dependent)

The number of lines in a text window in the *visual* command. The default is 8 at slow speeds (600 baud or less), 16 at medium speed (1200 baud), and the full screen (minus one line) at higher speeds.

w300, w1200, w9600

These are not true options but set *window* only if the speed is slow (300), medium (1200), or high (9600), respectively. They are suitable for an EXINIT and make it easy to change the 8/16/full screen rule.

wrapscan, ws (default: ws)

Searches that use regular expressions in addressing will wrap around past the end of the file.

wrapmargin, wm (default: wm=0)

Defines a margin for automatic wrapover of text during input in *open* and *visual* modes.

writeany, wa (default: nowa)

Inhibit checks normally made before write commands, allowing a write to any file which the system protection mechanism will allow.

307-262
Issue 1