2. VISUAL COMMAND MODE

Most visual mode commands are one keystroke long. The following tables list the operation performed by each keystroke, and also denotes any options or arguments that it accepts.

In addition to the keys listed here, your keyboard's "arrow" keys will be interpreted as the appropriate cursor movement commands. The same goes for (PgUp) and (PgDn), if your keyboard has them. The (Insert) key will toggle between insert mode and replace mode. There is a colon mode command (":map", to be described later) which will allow you to define other keys, such as function keys.

A tip: visual command mode looks a lot like text input mode. If you forget which mode you're in, just hit the (Esc) key. If Elvis beeps, then you're in visual command mode. If Elvis does not beep, then you were in input mode, but by hitting (Esc) you will have switched to visual command mode. So, one way or another, after (Esc) Elvis will be ready for a command.

2.1 Notation

The following notation is used in the tables that follow...

count
Many commands may be preceded by a count. This is a sequence of digits representing a decimal number. For most commands that use a count, the command is repeated *count* times, but a few commands use the count value for some completely different purpose. The count is always optional, and usually defaults to 1.

key
Some commands require two keystrokes. The first key always determines which command is to be executed. The second key is used as a parameter to the command.

mv
Some commands (! < > = c d y) operate on text between the cursor and some other position. Usually, the other position is specified by typing a movement command after the operator command, but there are other options. See the section on operators for more information.

inp
Many commands allow the user to interactively enter text. See the discussion of "input mode" in the following chapter.

2.2 Visual Commands, Grouped by Function

2.2.1 Edit commands

COMMAND	DESCRIPTION
P	Paste text before the cursor
p	Paste text after the cursor
count J	Join lines, to form one big line
count X	Delete the character(s) to the left of the cursor
count x	Delete the character that the cursor's on
count ~	Switch a character between uppercase & lowercase
count r key	Replace "count" chars by a given character
R inp	Overtype
count a inp	Insert text after the cursor
count A inp	Append at end of the line
count i inp	Insert text at the cursor
count I inp	Insert at the front of the line (after indents)
count o inp	Open a new line below the current line
count O inp	Open up a new line above the current line
count .	Repeat the previous "edit" command
count u	Undo the previous edit command
count ^R	Redo commands which were undone by the u command
U	Undo all recent changes to the current line

P
p
The **P** and **p** commands paste text from a cut buffer. The difference between them is that **p** pastes after the cursor, and **P** pastes before it. See the section on cut buffers for more information.

count **J**
The **J** command joins the current line with the following line. If you supply a count argument, then it will joint that many lines together, starting at the current line.

count **X**
count **x**
The **X** and **x** commands delete characters. The difference between them is that **x** deletes the character at the cursor, and **X** deletes the character before the cursor. If you supply a count, then it will delete that many characters. The deleted characters are copied into a cut buffer. The **X** and **x** commands never delete

newline characters.

count **~**
The **~** command changes uppercase letters to lowercase, or vice versa, and moves the cursor forward. Non-letters are unaffected, but the cursor will still be moved forward.

count **r** *key*
The **r** command replaces a single character in the edit buffer with a single character read from the keyboard. If you supply a count, then it will replace that many characters in the edit buffer with multiple copies of a single character from the keyboard.

R *inp*
The **R** command switches the window to "replace mode" which is a variation of input mode.

count **a** *inp*
The **a** command switches to input mode with the cursor starting immediately after its previous position. If a count is supplied, then after you manually insert the first copy of the new text, elvis will automatically insert *count-1* additional copies.

count **A** *inp*
The **A** command switches to input mode with the cursor starting at the end of the current line. If a count is supplied, then after you manually insert the first copy of the new text, elvis will automatically insert *count-1* additional copies.

count **i** *inp*
The **i** command switches to input mode with the cursor starting at its current position. If a count is supplied, then after you manually insert the first copy of the new text, elvis will automatically insert *count-1* additional copies.

count **I** *inp*
The **I** command switches to input mode with the cursor starting at the beginning of the current line. If a count is supplied, then after you manually insert the first copy of the new text, elvis will automatically insert *count-1* additional copies.

count **o** *inp*
The **o** command switches to input mode. A new, blank line is created after the current line, and the cursor starts at the beginning of that new line. If a count is supplied, then after you manually insert the first copy of the new text, elvis will automatically insert *count-1* additional copies.

count **O** *inp*
The **O** command switches to input mode. A new, blank line is created before the current line, and the cursor starts at the beginning of that new line. If a count is supplied, then after you manually insert the first copy of the new text, elvis will automatically insert *count-1* additional copies.

count .
The . command repeats the previous command which changed text. If a count is supplied, it overrides count of the original command.

count **u**
The **u** command undoes the previous edit command. If a count is supplied, then it will undo that many changes, within the limit defined by the undolevels option.

count **^R**
The **^R** command redoes commands which were undone by the **u** command. Essentially it undoes the "undo".

U
The **U** command undoes all changes which were made to the current line since the cursor was moved onto it.

2.2.2 Edit commands which are operators

COMMAND	DESCRIPTION
< mv	Shift text left
> mv	Shift text right
! mv text	Run selected lines thru an external filter program
= mv	Reformat
c mv inp	Change text
d mv	Delete text
y mv	Yank text (copy it into a cut buffer)

These commands all affect text between the cursor's current position and some other position. There are three ways that you can specify that other position:

* Follow the command keystroke with a movement command. For example, **dw** deletes a single word. **d3w** and **3dw** both delete three words.
* Type the command keystroke twice. This causes whole lines to be acted upon. For example, **>>** indents the current line. **3>>** indents the current line and the following two lines.
* Move the cursor to one end of the text, type **v**, **V**, or **^V** to start marking, move the cursor to the other end, and then type the desired command key.

< *mv*
> *mv*
These commands adjust the indentation of lines. The **<** command reduces indentation by shiftwidth columns, and the **>** command increases indentation by the same amount. The type of whitespace used for the new indentation is determined by the autotab option. The doubled-up **<<** and **>>** commands reduce or increase the indentation for the current line, respectively.

! *mv text*
This command prompts you to enter a shell command line at the

bottom of the window. When you've entered the command line, that command is executed and the selected text is piped through it. The resulting text replaces the original selected text. For example, **10!!sort** will send 10 lines through the sort program.

= *mv*

When applied to lines, this command resembles the **!** command, except that instead of prompting for a command line, the **=** command will always execute the program specified via the equalprg option. If you expect to be running the same filter program over and over again, then **=** is more convenient.

When applied to characters selected via the v command, the **=** command uses elvis' built-in calculator to evaluate the characters, and then replaces the original text with the result of that evaluation. For example, if you move the cursor to the start of a parenthesized arithmetic expression and hit **v%=** then the expression will be simplified.

c *mv inp*

This command puts you in input mode so you can replace the selected text with new, manually-entered text. The doubled-up **cc** command changes the current line.

d *mv*
y *mv*

These commands copy the selected text into a cut buffer. The **d** command then deletes the text from the edit buffer, but **y** leaves the original text intact. The doubled-up **dd** and **yy** commands delete or yank the current line, respectively.

2.2.3 Edit commands which are shortcuts for operators

COMMAND	DESCRIPTION
C inp	Change text from the cursor to the end of the line
D	Delete text from the cursor to the end of the line
count S inp	Change lines, like "count" cc
count s inp	Replace characters, like "count" cl
count Y	Yank text line(s) (copy them into a cut buffer)

C *inp*
D
count **S** *inp*
count **s** *int*
count **Y**

All of these commands are shortcuts for particular operator/movement command combinations. **C** is short for **c$**, **D** is short for **d$**, uppercase **S** is short for **cc**, lowercase **s** is short for **cl**, and **Y** is short for **yy**.

2.2.4 Movement commands which move by a line or column

COMMAND	DESCRIPTION
0	If not part of count, move to 1st char of this line
^	Move to the front of the current line (after indent)
$	Move to the rear of the current line
count |	Move to column "count" (defaulting to column 1)
count ^X	Move to column "count" (defaulting to the right edge)
count ^M	Move to the front of the next line
count +	Move to the front of the next line
count -	Move to the front of the preceding line
count G	Move to line #"count" (default is the bottom line)
count h	Move left
count ^H	Move left
count l	Move right
count Space	Move right
count j	Move down
count ^J	Move down
count ^N	Move down
count k	Move up
count ^P	Move up
count _	Move to the current line

0

The **0** command moves the cursor to the first character in the current line.

^

The **^** command moves the cursor to the first non-whitespace character in the current line. If the current line doesn't contain any non-whitespace characters, then the cursor is moved to the end of the line.

$

The **$** command moves the cursor to the last character in the line.

count **|**
count **^X**

These commands move the cursor to a given column. The leftmost column is designated "column 1." If the cursor can't be moved to the exact column number requested (e.g., because the line isn't that long, or the requested column falls in the middle of a tab character), then elvis will move the cursor as close as possible. If no count is given, then the **|** command moves the cursor to column 1 and the **^X** move the cursor to the rightmost visible column (taking side scrolling into account).

count **^M**
count **+**

The **^M** and **+** commands move the cursor down *count* lines (or 1 line if no count is given), and then to the first non-whitespace character in that destination line. It is equivalent to a **j** command followed by a **^** command.

count **-**
The **-** command moves the cursor up *count* lines (or 1 line if no count is given), and then to the first non-whitespace character in that destination line. It is equivalent to a **k** command followed by a **^** command.

count **G**
The **G** command moves the cursor directly to the start of a given line, or to the last line if no count is given.

For the purposes of this command, the "line number" of the cursor position is defined to be one plus the number of newline characters which precede it in the buffer. This definition is used regardless of what display mode you happen to be using. The number and ruler options use the same definition.

count **h**
count **^H**
The **h** and **^H** commands move the cursor *count* characters to the left, or 1 character leftward if no count is given. They won't move the cursor past the beginning of the line.

count **l**
count **Space**
The **h** and **Space** commands move the cursor *count* characters to the right, or 1 character rightward if no count is given. They won't move the cursor past the end of the line.

count **j**
count **^J**
count **^N**
These commands all move the cursor down *count* lines (or 1 line if no count is given), without changing the column of the cursor if possible.

count **k**
count **^P**
These commands all move the cursor up *count* lines (or 1 line if no count is given), without changing the column of the cursor if possible.

count **_**
The **_** command moves the cursor down *count-1* lines. This command is used internally to implement the double-operators; for example, << is interpreted as **<_**. By itself the **_** command is pretty useless.

2.2.5 Movement commands which are window-relative

COMMAND	DESCRIPTION
count H	Move to home row (the line at the top of the screen)
M	Move to middle row
count L	Move to last row (i.e., line at bottom of window)

count **H**
The **H** command moves the cursor to the front of the first line that is currently visible in the window. If a count is given, then it will move down *count-1* lines from the top of the window.

M
The **M** command moves the cursor to the front of the line in the middle of the window.

count **L**
The **L** command moves the cursor to the front of the line line that is currently visible in the window. If a count is given, then it will move up *count-1* lines from the bottom of the window.

2.2.6 Movement commands which search for strings

COMMAND	DESCRIPTION
/ text	Search forward for a given regular expression
? text	Search backward for a given regular expression
^A	Search for next occurrence of word at cursor
n	Repeat the previous search
N	Repeat previous search, but in the opposite direction

All of these search commands are affected by the magic, ignorecase, wrapscan, and autoselect options.

/ *text*
? *text*
These commands prompt you to enter a regular expression at the bottom of the window. When you have entered it, elvis will search forward (for **/**) or backward (for **?**) for text which matches the regular expression. Normally the cursor is positioned at the start of the matching text. If you want to include a **/** or **?** character in the regular expression, you must precede it with a backslash; otherwise it will be interpreted as a closing delimiter.

When entering the regular expression, you can append a closing delimiter (**/** or **?**, as appropriate) followed by a line delta (**+** or **-** followed by a line number) to move the cursor to the start of a line after or before the matching text. You can also use **v** or **n** flags to temporarily force the autoselect option on or off for that one search.

^A
This command searches forward for the next instance of the word under the cursor. It is a shortcut for the **/\<\@\> Enter** command.

n
N
These commands repeat the previous **/**, **?**, or **^A** command. The **n** command repeats it in the same direction as the original search, and **N** repeats it in the opposite direction.

2.2.7 Movement commands which search for characters

COMMAND	DESCRIPTION
count %	Move to matching (){}[] or to a given % of file
count F key	Move leftward to a given character
count f key	Move rightward to a given character
count T key	Move leftward *almost* to a given character
count t key	Move rightward *almost* to a given character
count ,	Repeat the previous [fFtT] but in the other direction
count ;	Repeat the previous [fFtT] cmd

count %
This command actually performs one of two totally separate movements, depending on whether a *count* was supplied. With no *count*, if the cursor is on a parenthesis character from the matchchar list, then it moves the cursor to the opposite parenthesis. If the cursor isn't on a parenthesis to begin with, then elvis will scan forward on the current line until it finds one, and then move to its opposite. It can also show matching preprocessor directives, if the window is in the syntax display mode for a language which has a preprocessor.

However, if a count is supplied, then it is used as a percentage from 1 to 100, and the cursor is moved to that percentage of the way into the buffer. For example, typing **50%** will move the cursor to the middle of the buffer.

count **F** *key*
count **f** *key*
These commands search for the character *key* in the current line, starting from the cursor position. For example, **fk** searches forward for the next "k". **F4** searches backward for the previous "4". If the end of line is reached without finding the character, then the command fails.

count **T** *key*
count **t** *key*
These commands move almost, but not quite, to the next instance of *key.* The **t** and **T** commands stop one character short of where **f** and **F** would stop, respectively.

count ,
count ;
The ; command repeats the previous **f**, **F**, **t**, or **T** command. You don't need to type a *key* because elvis automatically uses whatever *key* you looked for last time. The , command also

repeats the previous **f**, **F**, **t**, or **T** command, but in the reverse direction.

2.2.8 Movement commands which move by words

COMMAND	DESCRIPTION
count w	Move forward "count" words
count e	Move forward to the end of the current word
count b	Move back "count" words
count W	Move forward "count" Words
count E	Move end of Word
count B	Move back Word

The uppercase and lowercase versions of these commands differ only in their definition of a "word." The uppercase commands consider a word to be any sequence of non-whitespace characters, bound by whitespace characters or the ends of the buffer.

The lowercase commands define a word as either a sequence of alphanumeric characters, or a sequence of punctuation characters, but not a mixture of the two; these words can be bound by whitespace, the ends of the buffer, or by characters from the other class of lowercase word. (I.e, an alphanumeric word can be bound by punctuation characters, and a punctuation word can be bound by alphanumeric characters.) The underscore character is considered to be alphanumeric.

count **w**
count **W**
The **w** and **W** commands move the cursor forward to the start of the next word. If a count is given, the cursor will move forward *count* words.

count **e**
count **E**
The **e** and **E** commands move the cursor forward to the end of the current word, or if it is already at the end of the word then it'll move to the end of the following word. If a count is given, the cursor will move forward *count* word ends.

NOTE: These commands are often used as the targets of the operator commands, as in **dw** or **de**. When used this way, the difference between them is that **dw** includes any whitespace after the word, but **de** does not.

count **b**
count **B**
The **b** and **B** commands move the cursor backward to the start of the current word, or if it is already at the start of the word then it'll move to the start of the preceding word. If a count is given, the cursor will move backward *count* word starts.

2.2.9 Movement commands which move by sentences or sections

COMMAND	DESCRIPTION
count (	Move backward "count" sentences
count)	Move forward "count" sentences
count {	Move back "count" paragraphs
count }	Move forward "count" paragraphs
[[	Move back 1 section
]]	Move forward 1 section

count **(**
count **)**

These commands move backward or forward to the start of a sentence. The start of a sentence is defined to be the first non-whitespace character in the file, or the first non-whitespace character after the end of a sentence.

The exact definition of the end of a sentence depends on the values of the sentenceend, sentencegap, and sentencequote options. The default values of those options define a sentence end to be a period, question mark, or exclamation mark, followed by either a newline character, or two or more space characters. Any number of double-quote characters or closing parentheses may appear between the punctuation and the whitespace.

count **{**
count **}**

These commands move backward or forward to the start of a paragraph. The start of a paragraph is defined to be the first blank line encountered after a non-blank line; or a line which contains a troff paragraph command listed in the paragraphs option's value; or the start of a section as described below.

[[
]]

These commands move backward or forward to the start of a section. The start of a section is defined to be '{' character in column 1 of a line; or a troff section command listed in the value of the sections option.

Note: There are also [key and]key commands for recording keystrokes. Those commands are not related to the **[[** and **]]** movement commands.

2.2.10 Movement/mark commands

COMMAND	DESCRIPTION
m key	Mark a line or character
' key	Move to a marked line
` key	Move to a marked character
V	Start marking lines for c d y < > or !

v	Start marking characters for c d y < > or !
^V	Start marking a rectangle for c d y < > or !

m *key*
The **m***key* command stores the current cursor position in mark named *key,* where *key* is any letter. The mark is not visible.

' *key*
` *key*
These commands move the cursor back to a position which was stored via the **m***key* command. The `*key* (grave) command is a character-movement command which positions the cursor exactly on the marked character. The '*key* (apostrophe) command is a line-movement command which positions the cursor at the front of the line containing the mark.

The distinction between character-movement and line-movement becomes more significant when you're using the movement command as the target of an operator command. For example, after an **ma** command and some cursor movement, the command **d'a** would delete whole lines, but the command **d`a** would delete only the text between the cursor and the "a" mark.

v
V
^V
These commands start visually selecting text for use by an operator command. The lowercase **v** command starts selecting characters, uppercase **V** starts selecting lines, and **^V** starts selecting a rectangular area.

2.2.11 Scrolling commands

COMMAND	DESCRIPTION
^F	Move toward the bottom of the file by 1 screen full
^B	Move toward the top of the file by 1 screen full
z key	Scroll current line to window's +top -bottom .middle
count ^D	Scroll forward "count" lines (default 1/2 screen)
count ^E	Scroll forward "count" lines (default 1 line)
count ^U	Scroll backward "count" lines (default 1/2 screen)
count ^Y	Scroll backward "count" lines (default 1 line)

^F
This command moves forward one screen. Specifically, it locates the line at the bottom of the window and moves it to the top of the window. This means that for a 24-row window, ^F will generally move forward 22 lines. The cursor is moved if necessary to keep it on the screen.

^B

This command moves backward one screen, by moving the line at the top of the window to the bottom. It is the opposite of the ^F command.

z *key*
This command scrolls to bring the current line to either the top (for **z+**) middle (for **z.**) or bottom (for **z-**) of the window. You can also precede this command with a line number, in which case the cursor is moved to that line before the scrolling takes place; e.g., 98z+ will move the cursor to line 98 and then scroll as necessary to bring 98 to the top of the window.

Elvis also supports **zH**, **zM**, and **zL** as synonyms for those commands. These may be easier to remember, because they are somewhat analogous to the H, M, and L commands. In addition, **zz** is an easy-to-type synonym for scrolling a line to middle of the window.

count **^D**
count **^E**
These commands scroll the window forward by *count* lines. If no *count* is given, then **^E** defaults to 1 line, and **^D** defaults to the value of the scroll option. Supplying a *count* to the **^D** option has the side-effect of setting the scroll option to *count.*

count **^U**
count **^Y**
These commands scroll the window backward by *count* lines. If no *count* is given, then **^Y** defaults to 1 line, and **^U** defaults to the value of the scroll option. Supplying a *count* to the **^U** option has the side-effect of setting the scroll option to *count.*

2.2.12 Window commands

COMMAND	DESCRIPTION
^W s	Split current window
^W]	Split window, then look up tag at cursor
^W n	Split window, and create a new buffer
^W q	Save buffer & close window, like ZZ
^W c	Hide buffer & close window
^W d	Toggle the display mode
^W S	Toggle the sidescroll option
^W j	Move down to next window
^W k	Move up to previous window
count ^W ^W	Move to next window, or to the "count" window

^W s
This creates a new window, showing the same buffer as the current window. It is equivalent to the ex :split command with no arguments.

^W]
This performs tag lookup on the word at the cursor, and then creates a new window to show the tag definition. It is similar to the ex :stag command.

^W n
This creates a new buffer, and then creates a new window to show that buffer. It is similar to the ex :snew command.

^W q
This saves the buffer if it has changed, and then closes the window. It is identical to the visual ZZ command, and similar to the ex :xit command.

^W c
This closes the window, but it neither saves nor discards the current buffer. The buffer continues to exist. This command is like the ex :close command.

^W d
This toggles the windows display mode between two different modes. If the buffer's bufdisplay option is set to "html", "man", or "tex", then it will toggle between that mode and the syntax coloring mode. If bufdisplay is set to a syntax-coloring mode or "hex" then it toggles between that mode and "normal". If bufdisplay is "normal", then this command toggles between that mode and the "hex" mode.

^W S
This toggles the wrap option off or on, causing sideways scrolling to be alternately disabled and enabled.

^W j
^W k
count **^W ^W**
These commands make other windows be current. This is the only possible way to switch windows for some user interfaces such as the termcap interface.

NOTE: In addition to the commands shown here, some user interfaces may support extensions to these commands. For example, the termcap interface uses **^W+** to increase the size of the current window, **^W-** to reduce the size of the current window, and **^W\** to make the current window as large as possible.

2.2.13 Other commands

COMMAND	DESCRIPTION
" key	Select which cut buffer to use next
@ key	Execute the contents of a cut-buffer as VI commands
[key	Start recording keystrokes into a cut-buffer
] key	Stop recording keystrokes into a cut-buffer
: text	Run single EX cmd

Q	Quit to EX mode
K	Run keywordprg on the word at the cursor
Z Z	Save the file & exit
^Z	Either suspend elvis, or fork a shell
^G	Show file status, and the current line #
^L	Redraw the screen
*	Go to the next error in the errlist
^I	In HTML mode, move forward to next link
^T	Return to source of previous :tag or ^] command.
^]	If the cursor is on a tag name, go to that tag
^^	Switch to the previous file, like ":e #"
count # key	Increment a number
count &	Repeat the previous ":s//" command here

" *key*
This command determines which cut buffer the next command will use. If the next command doesn't involve a cut buffer, then this command has no effect. If you don't use this command, then the following command will use the anonymous cut buffer.

@ *key*
The **@** command executes the contents of a cut buffer as a series of vi command keystrokes. The key after the **@** is the name of the cut buffer to be executed.

[*key*
] *key*
These commands are used for recording keystrokes into a cut-buffer. The *key* is the name of the cut buffer; it must be a letter. Later, you can replay the keystrokes via the **@***key* command.

Note: There are also [[and]] commands for moving the cursor to the start of a section. Those movement commands are unrelated to these **[***key* and **]***key* keystroke recording commands.

:
Q
This : command allows you to enter a single ex command, and then immediately return to visual command. The **Q** command is similar, except that it causes you to remain in ex mode until you give the ex :vi command.

K
This command executes the program named in the keywordprg option, passing it the word at the cursor. This is intended to provide an easy way to run on-line manuals and the like.

ZZ
The **ZZ** command writes the current file if it has changed, and then exits. It is equivalent to the ex :xit command.

^Z
This command will either suspend elvis, or fork a shell program.

Either way, you'll get a shell prompt. **^Z** is equivalent to the :stop ex command.

^G
The **^G** command displays the status of the current file. It is equivalent to the ex :file command with no arguments.

^L
The **^L** command causes the current window to be redrawn from scratch. This is handy when another program or line noise changes the screen in a way that elvis can't detect & correct automatically.

*****
The ***** command moves the cursor to the next error reported by the compiler. It is equivalent to the ex :errlist command.

^I
The **^I** command (the **Tab** key) moves the cursor forward to the next hypertext link, if you're in the "html" display mode.

^]
^T
The **^]** command performs tag lookup on the word at the cursor position, and moves the cursor to the file/line where the tag is defined. It resembles the ex :tag command. The **^T** command returns the cursor to the previous position, like :pop.

^^
After you've switched edit buffers in a window, the **^^** (Control-Carat) command switches back to the previous buffer, and moves the cursor to its previous position within that buffer. It is like the ex :e# command.

count **#** *key*
This command allows you add or subtract *count* to the number at the cursor. If the cursor isn't on a word that looks like a number, then the command fails. If no *count* is given, then 1 is assumed. If the *key* is "-" then *count* is subtracted from the number. If the *key* is "+" or "#" then *count* is added to the number. If the *key* is "=" then the word is changed to *count*. Any other *key* causes the command to fail.

count **&**
Repeat the previous :s/re/new/ command on the current line. If *count* is given, then also apply it to each of the *count-1* following lines as well.

2.3 Visual Commands, sorted by their ASCII code

COMMAND	DESCRIPTION

	NUL	(undefined)
	^A	Search for next occurrence of word at cursor
	^B	Move toward the top of the file by 1 screen full
	^C	(undefined; may abort a time-consuming command)
count	^D	Scroll forward "count" lines (default 1/2 screen)
count	^E	Scroll forward "count" lines (default 1 line)
	^F	Move toward the bottom of the file by 1 screen full
	^G	Show file status, and the current line #
count	^H	Move left
	^I	(Tab) In "html" display mode, move to next hyperlink
count	^J	Move down
	^K	(undefined)
	^L	Redraw the screen
count	^M	Move to the front of the next line
count	^N	Move down
	^O	ignored, to simplify implementation of "visual" map
count	^P	Move up
	^Q	(undefined; may resume stopped output)
count	^R	Redo commands which were undone by the u command
	^S	(undefined; may stop output)
	^T	Return to source of previous :tag or ^] command.
count	^U	Scroll backward "count" lines (default 1/2 screen)
	^V	Start marking a rectangle for c d y < > or !
count	^W ^W	Move to next window, or to the "count" window
	^W S	Toggle the sidescroll option
	^W]	Split window, then look up tag at cursor
	^W c	Hide buffer & close window
	^W d	Toggle the display mode
	^W j	Move down to next window
	^W k	Move up to previous window
	^W n	Split window, and create a new buffer
	^W q	Save buffer & close window, like ZZ
	^W s	Split current window
count	^X	Move to column "count" (defaulting to the right edge)
count	^Y	Scroll backward "count" lines (default 1 line)
	^Z	Either suspend elvis, or fork a shell
	^[	(Escape) Cancels a partially-entered command
	^\	(undefined; may cause core dump)
	^]	If the cursor is on a tag name, go to that tag
	^^	Switch to the previous file, like ":e #"
	^_	(undefined)
count	Space	Move right
	! mv	Run selected lines thru an external filter program
	" key	Select which cut buffer to use next
count	# key	Increment a number
	$	Move to the rear of the current line
count	%	Move to matching (){}[] or to a given % of file
count	&	Repeat the previous ":s//" command here
	' key	Move to a marked line
count	(	Move backward "count" sentences
count	)	Move forward "count" sentences
	*	Go to the next error in the errlist
count	+	Move to the front of the next line
count	,	Repeat the previous [fFtT] but in the other direction
count	-	Move to the front of the preceding line
count	.	Repeat the previous "edit" command

	/ text	Search forward for a given regular expression
	0	If not part of count, move to 1st char of this line
	1	Part of a count argument
	2	Part of a count argument
	3	Part of a count argument
	4	Part of a count argument
	5	Part of a count argument
	6	Part of a count argument
	7	Part of a count argument
	8	Part of a count argument
	9	Part of a count argument
	: text	Run single EX cmd
count	;	Repeat the previous [fFtT] cmd
	< mv	Shift text left
	= mv	Reformat
	> mv	Shift text right
	? text	Search backward for a given regular expression
	@ key	Execute the contents of a cut-buffer as VI commands
count	A inp	Append at end of the line
count	B	Move back Word
	C inp	Change text from the cursor to the end of the line
	D	Delete text from the cursor to the end of the line
count	E	Move end of Word
count	F key	Move leftward to a given character
count	G	Move to line #"count" (default is the bottom line)
count	H	Move to home row (the line at the top of the screen)
count	I inp	Insert at the front of the line (after indents)
count	J	Join lines, to form one big line
	K	Run keywordprg on the word at the cursor
count	L	Move to last row (i.e., line at bottom of window)
	M	Move to middle row
	N	Repeat previous search, but in the opposite direction
count	O inp	Open up a new line above the current line
	P	Paste text before the cursor
	Q	Quit to EX mode
	R inp	Overtype
count	S inp	Change lines, like "count" cc
count	T key	Move leftward *almost* to a given character
	U	Undo all recent changes to the current line
	V	Start marking lines for c d y < > or !
count	W	Move forward "count" Words
count	X	Delete the character(s) to the left of the cursor
count	Y	Yank text line(s) (copy them into a cut buffer)
	Z Z	Save the file & exit
	[[	Move back 1 section
	[key	Start recording keystrokes into a cut-buffer
	\	(undefined)
	]]	Move forward 1 section
	] key	Stop recording keystrokes into a cut-buffer
	^	Move to the front of the current line (after indent)
count	_	(the underscore character) Move to the current line
	` key	Move to a marked character
count	a inp	Insert text after the cursor
count	b	Move back "count" words
	c mv	Change text
	d mv	Delete text

count e	Move forward to the end of the current word
count f key	Move rightward to a given character
g	(undefined)
count h	Move left
count i inp	Insert text at the cursor
count j	Move down
count k	Move up
count l	Move right
m key	Mark a line or character
n	Repeat the previous search
count o inp	Open a new line below the current line
p	Paste text after the cursor
q	(undefined)
count r key	Replace "count" chars by a given character
count s inp	Replace characters, like "count" cl
count t key	Move rightward *almost* to a given character
count u	Undo the previous edit command
v	Start marking characters for c d y < > or !
count w	Move forward "count" words
count x	Delete the character that the cursor’s on
y mv	Yank text (copy it into a cut buffer)
z key	Scroll current line to window’s +top -bottom .middle
count {	Move back "count" paragraphs
count |	Move to column "count" (defaulting to column 1)
count }	Move forward "count" paragraphs
count ~	Switch a character between uppercase & lowercase
DEL	(undefined)