vi Cheat Sheet

STATIONX
THE CYBER SECURITY COMPANY

What Is vi?

The command-line text editor vi (short for “visual interface”) is a standard utility on Unix/Linux, and it comes pre-installed on Unix. It contains multiple modes, the basic ones being: **Command** mode (the default mode; unless otherwise specified, the commands in this cheat sheet apply to this mode), **Insert** mode, and **Visual** mode.

This vi editor cheat sheet contains specific commands for various functions, and all commands are case-sensitive (capital letter command = Shift + original letter key).

Some Unix/Linux distributions use vi as an alias for **vim**, another command-line text editor which is an improvement on vi in that it has productivity-enhancing features, such as window-splitting and tabs, code highlighting, macros, multiple-time undo’s and redo’s, command-line history, pattern matching, and word completion.

Pro tip: To execute a vi command multiple times, prefix the command with a positive integer. For example, if you want to delete 100 lines, type “`100dd`” in Command mode.

Basic Navigation

This section covers opening vi; moving the cursor around; jumping to the start or end of a word, line, paragraph, and file; and searching for text patterns. The <Return> key featured in some vi commands below is the same as the <Enter> key on some keyboards. <Ctrl> is the Ctrl key, and <Escape/ESC> is the Escape key. Other instances of < and > are literal.

Here are some helpful commands for entering/using vi on the command line:

Terminal command	Explanation

`vi filename.txt`	Open a new or existing file called filename.txt
`vi -r filename.txt`	Recover a file called filename.txt that someone was editing when the operating system crashed
`view filename.txt`	Display read-only filename.txt
`cat filename.txt`	Output contents of filename.txt; suitable for small files
`less filename.txt`	Output contents of filename.txt; suitable for large files; navigate using arrow keys

On some distributions, such as macOS, you may use the arrow keys to move the cursor left-, right-, up-, and downwards in the default Command mode. However, on other Unix/Linux distributions, using the arrow keys might yield one of A, B, C, and D, so you still need to learn the following direction commands:

Direction command	Explanation
`h`	Move cursor leftwards by one character
`j`	Move cursor downwards by one line
`k`	Move cursor upwards by one line
`l`	Move cursor rightwards by one character
`12h`	Move cursor leftwards by 12 characters
`23j`	Move cursor downwards by 23 lines
`34k`	Move cursor upwards by 34 lines
`45l`	Move cursor rightwards by 45 characters
`100j`	Move cursor downwards by 100 lines

Use the following directional commands to jump to the beginning or end of a word (a string of alphanumeric characters excluding spaces and punctuation), line, paragraph, or file:

Command	Explanation
`b`	Move cursor to the beginning of the current word
`w`	Move cursor to the beginning of the next word
`e`	Move cursor to the end of the current word
`B`	Move cursor to the beginning of the previous word before a whitespace
`W`	Move cursor to the beginning of the next word after a whitespace
`E`	Move to the end of the current word before a whitespace
`0`	Move cursor to the first character in the current line
`^`	Move cursor to the beginning of the current line
`$`	Move cursor to the final character in the current line
`gg`	Go to the first line in the document
`` `` ```	Go to your last position in the file
`+`	Move cursor to beginning of next line
`-`	Move cursor to beginning of previous line

<Ctrl>d	Scroll down one-half screen
<Ctrl>u	Scroll up one-half screen
<Ctrl>f	Scroll forward one full screen
<Ctrl>b	Scroll backward one full screen
)	Move cursor to the next sentence
(	Move cursor to the previous sentence
{	Move backward by one paragraph
}	Move forward by one paragraph
H	Move to the top line of the screen
M	Move to the middle line of the screen
L	Move to the last line of the screen
%	Move to matching bracket: () [] {}
:0<Return>	Move cursor to the first line in the document
1G	Move cursor to the first line in the document
:n<Return>	Move cursor to the n-th line, where n is a positive integer, in the document e.g., :10<Return> moves the cursor to the tenth line
nG	Move cursor to the n-th line, where n is a positive integer, in the document e.g., 5G moves the cursor to the fifth line
G	Go to the final line in the document

A tricky part of mastering vi is searching for patterns and replacing them where needed. The following table lists the relevant search-and-replace vi commands:

Command	Explanation
/	Anything you type after this symbol becomes a pattern you want to find forwards/downwards in the file.
?	Anything you type after this symbol becomes a pattern you want to find backwards/upwards in the file.
/^string	Find the pattern string matching the beginning of a line
/string$	Find the pattern string matching the end of a line
n	Find the next occurrence of the pattern typed after /
N	Find the previous occurrence of the pattern typed after ?
f<char>	Find a character <char> (such as “a”, “0”, …) on the same line, moving forwards till the end of the line
F<char>	Find a character <char> (such as “a”, “0”, …) on the same line, moving backwards till the beginning of the line
;	Repeat the previous character search in the same direction
:s/foo/bar/ig	Replace all occurrences of “foo” with “bar” in the current line; “i” means “case-insensitive” and “g” stands for “global”

`:1,$s/foo/bar`	Replace an occurrence of "`foo`" with "`bar`" from the first line to the last line
`:11,22s/foo/bar/gI`	Replace all occurrences of "`foo`" with "`bar`" from the 11th line to the 22nd line; "`g`" stands for "global" and "`I`" means "case-sensitive"
`:^,.s/foo/bar/g`	Replace all occurrences of "`foo`" with "`bar`" from the beginning of the file to the current cursor position
`:%s/foo/bar`	Replace an occurrence of "`foo`" with "`bar`" in the document
`:%s/foo/bar/g`	Replace all occurrences of "`foo`" with "`bar`" in the document; "`g`" stands for "global"
`:%s/foo/bar/c`	Replace all occurrences of "`foo`" with "`bar`" in the document; "`c`" means vi will show a prompt to confirm each replacement ("Y" to confirm)
`:&`	Repeat the last replacement command
`/\<pro\>`	Search for the word `pro` (and not for `proper`, `produce`, etc.)
`/l[aei]nd`	Search for `land`, `lend`, and `lind`

Editing Text

Here, we cover Insert mode, the deletion, modification, and repetition of text, and undoing and redoing actions.

Insert mode is where you edit the text contents of a file. Once in Insert Mode, the word "INSERT" will appear along the bottom edge of the terminal. The following table shows several ways to enter Insert mode in vi:

Command to enter Insert mode	Explanation
`i`	Inserts text before the current cursor location
`a`	Inserts text after the current cursor location
`o`	Creates a new line for text entry below the cursor location
`I`	Inserts text at the beginning of the current line
`A`	Inserts text at the end of the current line
`O`	Creates a new line for text entry above the cursor location

The following commands help you delete content and enter Insert mode at the same time:

Command	Explanation
`cc`	Remove the contents of the current line. Afterward, vi remains in Insert mode.

`C`	Remove the contents of the current line. Afterward, vi remains in Insert mode.
`cw`	Remove the word indicated by the cursor. Afterward, vi remains in Insert mode.
`c11w`	Remove 11 words starting from the one indicated by the cursor. Afterward, vi remains in Insert mode.
`12cc`	Remove 12 lines starting from the one indicated by the cursor. Afterward, vi remains in Insert mode.
`c20c`	Remove 20 lines starting from the one indicated by the cursor. Afterward, vi remains in Insert mode.
`s`	Remove the current character. Afterward, vi remains in Insert mode.
`S`	Deletes the current line. Afterward, vi remains in Insert mode.

When you've finished modifying the text, use the following command to stop inadvertently editing your file:

Command to exit	Explanation
`<Escape/ESC>`	Exit Insert mode

Remember the following commands if you want to delete one or more characters, words, lines, or paragraphs:

Command	Explanation
`x`	Delete the character highlighted by the cursor
`X`	Delete the character before the cursor location
`dd`	Delete the line the cursor is on
`dw`	Deletes from the current cursor location to the next word
`dW`	Delete a blank-delimited word and the following space
`d}`	Delete all characters to the end of the paragraph
`:5,30d`	Delete lines 5–30
`3x`	Delete three characters starting from the one indicated by the cursor
`d9w`	Delete nine words starting from the one indicated by the cursor
`12dd`	Delete 12 lines starting from the one indicated by the cursor
`d20d`	Delete 20 lines starting from the one indicated by the cursor
`d^`	Delete from the current cursor position to the beginning of the line
`d$`	Delete from the current cursor position to the end of the line

`D`	Delete from the cursor position to the end of the current line
`dG`	Delete from the current line to the end of the file

The commands listed below are for changing characters/words/lines, repeating them, and undoing changes:

Command	Explanation
`u`	Undo previous action; repeat as often as is necessary
`U`	Undo all changes to the current line
`.`	Redo the last command once
`<Ctrl>r`	Redo the last command once
`n.`	Redo the last command `n` times, where `n` is a positive integer
`J`	Join next line to the current line
`xp`	Switch the positions of two adjacent characters
`ddp`	Swap two adjacent lines
`:15,16 co 17`	Copy lines 15–16 to after line 17
`:18,20 m $`	Move lines 18–20 to the end of the file
`:7,300 d`	Copy lines 7–300 to the buffer and delete them from the document

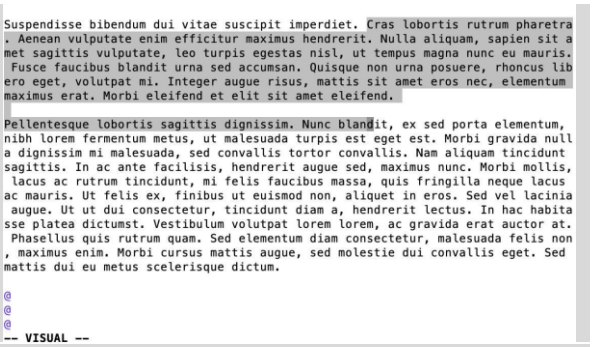
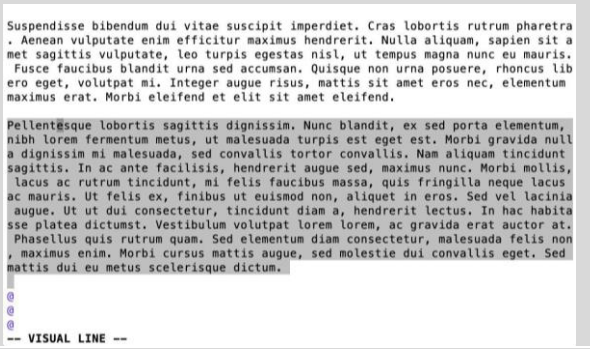
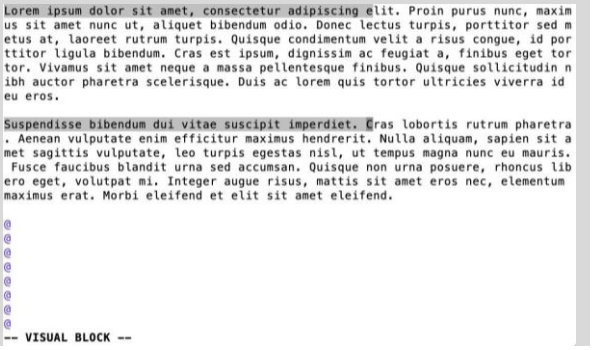
Visual Mode

The Visual mode in vi is for highlighting and selecting text. In this special mode, you can be precise in actions such as cutting, copying, pasting, making uppercase/lowercase, and replacing words.

Three Visual modes exist:

- visual character mode,
- visual line mode, and
- visual block mode.

Command to enter Visual mode	Explanation
`v`	Enter visual character mode; afterward, use the navigation keys to highlight text. Once in this mode, the word “VISUAL” will appear along the bottom edge of the terminal.

	
V	Enter visual line mode, highlighting the entire line on which the cursor is. Once in this mode, the word “VISUAL LINE” will appear along the bottom edge of the terminal. 
<Ctrl>v	Enter visual block mode, making text selections by blocks. Moving the cursor will make rectangle selections of the text. Once in this mode, the word “VISUAL BLOCK” will appear along the bottom edge of the terminal. 

Once in any of these modes, you can highlight the desired text using arrow keys or the navigation commands in vi. Afterward, you can delete, copy, paste, and manipulate the text wherever the cursor is using the following commands:

Command	Explanation
yy	Copy (y = yank to buffer) the current line indicated by the cursor

`40yy`	Copy 40 lines into the buffer, starting from the current line indicated by the cursor
`yw`	Copy the current word from the character the cursor is on to the end of the word
`:15,20y`	Copy lines 15–20
`p`	Paste the copied text after the cursor
`:put<Return>`	Put (paste) the copied text after the cursor
`P`	Paste the copied text before the cursor
`yyp`	Repeat the current line
`ywP`	Repeat the copied word
`r`	Replace the character highlighted by the cursor
`R`	Overwrite multiple characters beginning with the character currently under the cursor; stop replacing with <Escape/ESC>
`~`	Change the alphabetical character under the cursor between uppercase and lowercase
`>`	Increase indentation of all lines
`<`	Decrease indentation of all lines

Any copied/yanked content goes into one of 26 temporary memory receptacles in the vi editor called text buffers. They persist until you copy or delete more characters into it or until you quit your current vi session. The name of each text buffer is a letter of the English alphabet, so their names are `a` through `z`.

Here are some vi commands to manipulate vi text buffers:

Command	Explanation
`"ayy`	Copy the current line into buffer `a`
`"Ayy`	Append the current line to buffer a
`"add`	Delete the current line and put text in buffer `a`
`"ap`	Paste the line from buffer `a` below the current line
`"a100yy`	Copy 100 lines into buffer `a`
`"a100dd`	Copy 100 lines of text into buffer `a` and delete them from the document

The vi editor allows you to use abbreviations to replace words. After typing the abbreviation, you expand an abbreviation when you hit <Space> or <Return>. Abbreviations can be a life-saver as you can define common typos as demonstrated by the examples below:

Command	Explanation
`:ab os operating system`	Expand every newly typed instance of “os” into “operating system”
`:abbreviate sig your@email.com`	Expand every newly typed instance of “sig” into “your@email.com”
`:ab teh the`	Auto-correct “the” typo
`:ab adn and`	Auto-correct “and” typo
`:ab taht that`	Auto-correct “that” typo

`:iab tihs this`	Only auto-correct “this” typo in Insert mode
`:cab hoeewvr however`	Only auto-correct “however” typo in Command mode
`:abc`	Clear all abbreviations
`:abclear`	Clear all abbreviations
`:una os`	Remove the abbreviation related to “os”
`<Ctrl>v`	Prevent an abbreviation you’re entering at the moment from expanding
`<CR>`	This string of four characters represents the new line character when constructing an abbreviation in Insert mode. “CR” stands for “carriage return.” You can type similar special characters, such as `<tab>` to input a tab and `<esc>` to represent the <Escape> character.

When you’ve finished your work in Visual mode, press the Escape key twice:

Command	Explanation
`<Escape/ESC><Escape/ESC>`	Exit Visual mode

Command Mode

Command mode is the default mode you see when you enter vi. This section covers saving files, quitting the vi editor, showing and hiding line numbers, and running shell commands from inside vi.

Command	Explanation
`ZZ`	Save (if there are changes) and quit
`:w<Return>`	Save (write) to the file named in the original vi execution
`:q<Return>`	Quit (exit the vi console); this will only work if you’ve made no changes
`:wq<Return>`	Save and quit
`:q!<Return>`	Quit without saving
`:w newfilename.txt<Return>`	Save to `newfilename.txt`
`:w>>extrafile.txt<Return>`	Append the current file to a file named `extrafile.txt`
`:w!<Return>`	Overwrite the contents of vi to the file named in the original vi execution
`:w! newfilename.txt<Return>`	Overwrite the contents of vi to `newfilename.txt`
`:23,45w snippet.txt<Return>`	Write the contents of the lines numbered 23 through 45 to a new file named `snippet.txt`
`:23,45w>>snippet.txt<Return>`	Append the contents of the lines numbered 23 through 45 to a new file named `snippet.txt`

`:r filename.txt<Return>`	Read a file named `filename.txt` and insert its contents after the cursor in the currently opened file
`:h<Return>`	Get help on vi (exit it with `:q`)
`:set nu<Return>`	Display line numbers
`:set nonu<Return>`	Hide line numbers
`<Ctrl>g`	Show the current line number and the total number of lines in the file at the bottom of the screen
`:.=`	Return the line number where the cursor is at the bottom of the screen
`:=`	Return the total number of lines in the document at the bottom of the screen
`:!<shell_command><Return>`	Run a `<shell_command>`
`:!ls<Return>`	Run the `ls` (list items in current working directory) command from vi

Advanced Features

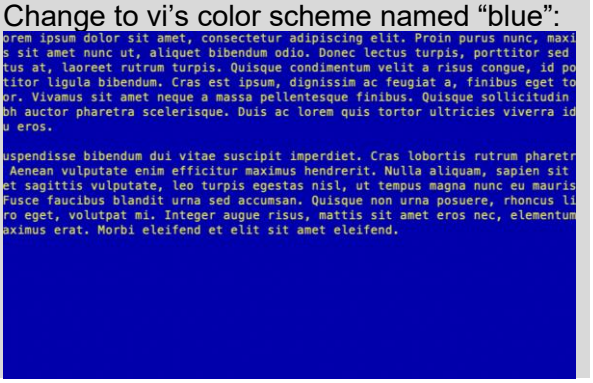
This part will cover regular expressions, customization of the vi interface, macros, and splitting the vi editor into multiple windows/screens.

The vi editor admits regular expressions as search strings.

Regular expression	Denote	Character class (where applicable)
`^`	The beginning of the line: use at the beginning of a search pattern.	/
`.`	Any single character except new line	/
`*`	Zero or more of the previous character	/
`$`	The end of the line: use at the end of the search pattern.	/
`[`	The beginning of a set of matching or non-matching search patterns	/
`]`	The end of a set of matching or non-matching search patterns	/
`\<`	The beginning of a word in a search pattern	/
`\>`	The end of a word in a search pattern	/
`\s`	whitespace character	<Space>, <Tab>
`\S`	non-whitespace character	All characters except <Space> and <Tab>
`\d`	digit	`[0-9]`
`\D`	non-digit	`[^0-9]`
`\x`	hex digit	`[0-9A-Fa-f]`
`\X`	non-hex digit	`[^0-9A-Fa-f]`
`\o`	octal digit	`[0-7]`
`\O`	non-octal digit	`[^0-7]`
`\h`	head of word character	`[A-Za-z_]`

\H	non-head of word character	[^A-Za-z_]
\p	printable character	[-~]
\P	printable character, excluding digits	(?![0-9])[-~]
\w	word character	[0-9A-Za-z_]
\W	non-word character	[^0-9A-Za-z_]
\a	alphabetic character	[A-Za-z]
\A	non-alphabetic character	[^A-Za-z]
\l	lowercase character	[a-z]
\L	non-lowercase character	[^a-z]
\u	uppercase character	[A-Z]
\U	non-uppercase character	[^A-Z]

To configure the look and feel of your vi editor, use the following commands:

Command	Explanation
:colorscheme <Ctrl>d	Show a list of available vi color schemes
:colo blue	Change to vi's color scheme named "blue": 

This article contains additional commands on setting your vi color scheme.

A vi macro is a feature that allows you to record a sequence of commands for performing a certain task. Multiple executions of that macro will repeat the same task in an automated fashion.

Macro command	Explanation
q<register><command(s)>q	The syntax for recording a macro. Examples below.
qao<ESC>q	Record a basic macro that inserts a new line (o) and save it to register a
:reg	View saved macros
@a	Replay the macro saved in register a
5@a	Execute the macro saved in register a on five more lines

You can also split your vi editor screen into multiple windows:

Command	Explanation
<Ctrl>ws	Split the screen horizontally

<Ctrl>wv	Split the screen vertically
<Ctrl>ww	Navigate between horizontal/vertical split screens
<ESC>:q	Exit one of the split screens

For advanced screen splits, refer to our tmux cheat sheet.

The following commands help you configure the settings for your vi user experience:

Setting command	Explanation
:set ic<Return>	Ignore the case when searching
:set ai<Return>	Set auto indent
:set noai<Return>	Unset auto indent
:set nu<Return>	Display lines with line numbers on the left side
:set sw = n<Return>	Set the shift width of a software tabstop to a length of n, where n is a positive integer
:set sw = 4<Return>	Set a shift width of four characters
:set ws<Return>	Allow your pattern searches to loop around
:set wm = 0<Return>	Turn off wrap margin
:set wm = n<Return>	Set the wrap margin from the right edge of the screen as the specified number of characters n, where n is a positive integer
:set wm = 2<Return>	Set the wrap margin to two characters
:set ro<Return>	Change file type to “read only”
:set term<Return>	Print terminal type
:set bf<Return>	Discard control characters from input
:set all<Return>	View a list of all settings and their current values
:set all&<Return>	Reset all settings to their default values

Conclusion

We hope this vi cheat sheet makes you a more confident user of vi commands and helps you complete your work more efficiently. Remember to check out our courses on Unix/Linux shell programming and articles on IT Fundamentals to fill in any proficiency gaps you have in Unix/Linux.