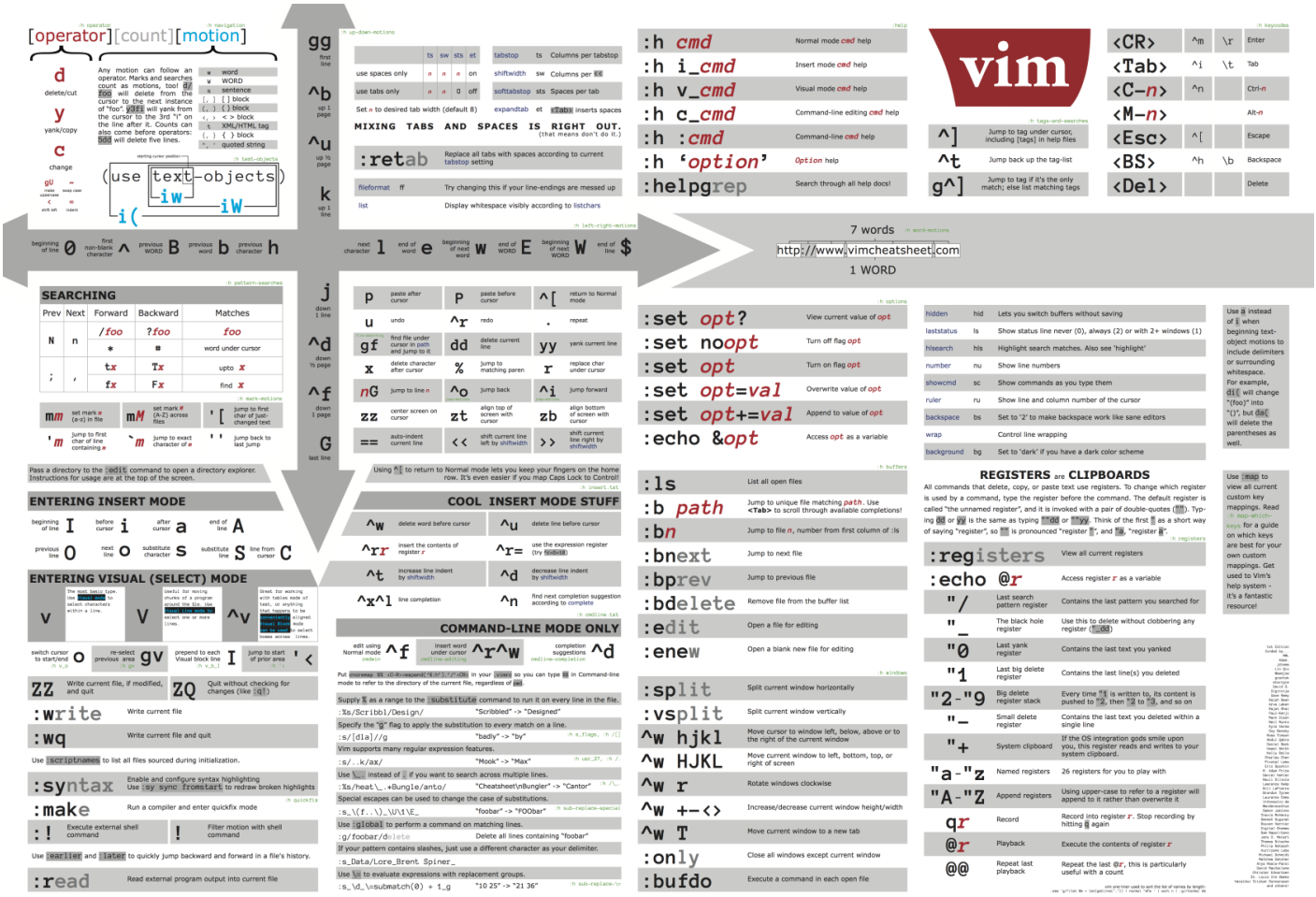
Vim Tips ??



http://www.maxcantor.net/

91 Vim Keyboard Shortcuts to Get Started with Vim - Tech Inscribed

Source: [91 Vim Keyboard Shortcuts to Get Started with Vim - Tech Inscribed](#).

This article gives you a list of most commonly used Vim keyboard shortcuts that can help you get started with Vim.

Modes

Vim has multiple modes namely Normal mode, Insert mode, Replace mode, and Visual mode.

Normal mode is where you can use most of the shortcuts. In insert mode, Vim behaves like a normal text editor. In replace mode, existing texts are replaced or overwritten as you type. Visual mode allows us to select texts visually and then make changes to it.

`i`	Insert mode at the cursor.
`I`	Insert mode at the beginning of the line.
`a`	Insert mode after the cursor.
`A`	Insert mode at the end of the line.
`o`	Insert mode with a new line below.
`O`	Insert mode with a new line above.
`s`	Insert mode at the cursor, after deleting the current character.
`S`	Insert mode, after deleting the current line.
`v`	Visual mode at the cursor.
`V`	Visual mode at the beginning of the line.
`r`	Replace mode to replace the current character.
`R`	Replace mode.
`Esc`	Normal Mode or Command Mode.

Motion or Movement

Motions are a set of shortcut keys that allow us to quickly move around the text document.

`k`	Move one line up.
`j`	Move one line down.
`h`	Move one line left.
`l`	Move one line right.
`w`	Go to the beginning of the next word (separated by space/punctuation).
`W`	Go to the beginning of the next word separated by space.

`e`	Go to the end of the next word separated by punctuations/space.
`E`	Go to the end of the next word separated by space.
`b`	Go to the beginning of the previous word separated by punctuation/space.
`B`	Go to the beginning of the previous word separated by space.
`{`	Go to the previous line break.
`}`	Go to the next line break.
`%`	Go to matching Bracket.
`#`	Go to the previous occurrence of the current word under the cursor.
`*`	Go to the next occurrence of the current word under the cursor.
`^`	Go to the first non-empty character in the line.
`0`	Go to the beginning of the line.
`$`	Go to the end of the line.
`gg`	Go to the beginning of the file.
`G`	Go to the end of the file.
`gd`	Go to definition.
`:{number}`	Go to the line number. Note: This is not really a shortcut but a Vim command.

Motion shortcuts can be prefixed with a number to repeat the motion. For example, to move 5 lines down, we can use `5j`. Similarly, to move 2 words forward, we can use `2w`.

Also Read: *How to Set up Vim as an IDE for React and TypeScript in 2020*

Searching

You can search for a character in the current line using find `f` and till `t`.

The difference between “find” and “till” is that “find” moves the cursor to the searched character, whereas “till” moves the cursor to the previous character of the searched character.

`f`	Find the next occurrence of a character in the current line and go to it.

`t`	Find the next occurrence of a character in the current line and go to its previous character.
`F`	Find the previous occurrence of a character in the current line and go to it.
`T`	Find the previous occurrence of a character in the current line and go to its next character.

To search for a phrase, you can use Vim command `/` or `?`

`/{search-word}`	Search for a word forward. For example `/export` will search and find the next instance of the word “export”.
`?{search-word}`	Search for a word backward. For example `/export` will search and find the previous instance of the word “export”.

After searching, `n` and `N` can be used to find next and previous occurrences respectively.

`n`	Find the next occurrence. To be used after using `/` or `?`
`N`	Find the previous occurrence. To be used after using `/` or `?`

Deleting

Delete(`d`) is an operator in Vim. Operators cannot function without motion and hence `d` is always followed by a motion. Here, the motion is what tells Vim what to delete.

That being said, you can combine all motions with `d`, like so.

`dw`	Delete from current character to end of a word(space/punctuation/EOL).
`dW`	Delete from current character to end of a word (space/EOL).
`db`	Delete from current character to beginning of a word(space/punctuation/EOL).
`dB`	Delete from current character to beginning of a word(space/EOL).
`diw`	Delete the current word.
`diW`	Delete the current word.
`dd`	Delete the current line.

`di'`	Delete everything within the single quotes.
`di"`	Delete everything within the double-quotes.
`di(`	Delete everything within the brackets.
`di{`	Delete everything within the curly braces.
`di[`	Delete everything within the square brackets.
`x`	Delete the current character under the cursor.
`X`	Delete the previous character.

You can also repeat these commands by prefixing a number. For example, you can delete 5 lines using `5dd` or `d4j`

Changing

In Vim, changing is similar to delete, the only difference is that after deleting, insert mode gets activated.

`cw`	Delete from current character to end of a word(space/punctuation/EOL) and then go to insert mode.
`cW`	Delete from current character to end of a word (space/EOL) and then go to insert mode.
`cb`	Delete from current character to beginning of a word(space/punctuation/EOL) and then go to insert mode.
`cc`	Change the current line.
`cB`	Change the current block.
`ciw`	Change inside a word.
`ciW`	Change inside a word.
`ci'`	Change everything inside a pair of single quotes.
`ci"`	Change everything inside a pair of double-quotes.
`ci(`	Change everything inside a pair of parentheses.
`ci{`	Change everything inside a pair of curly braces.
`ci[`	Change everything inside a pair of square brackets.
`s`	Delete the current character and go to insert mode.
`S`	Delete the current line and go to insert mode.
`>>`	Indent current line.

`<<`	Unindent current line.

Copy(Yank) and Paste

`yy`	Copy the current line
`yw`	Copy the current word from cursor till space/punctuation
`yW`	Copy the current word from cursor till space.
`yiw`	Copy the current word.
`yiB`	Copy the block.
`yi'`	Copy everything inside a pair of single quotes.
`yi"`	Copy everything inside a pair of double-quotes.
`yi(`	Copy everything inside a pair of parentheses.
`yi{`	Copy everything inside a pair of curly braces.
`yi[`	Copy everything inside a pair of square brackets.
`p`	Paste below the current line.
`P`	Paste above the current line.

When a text is yanked, it goes into Vim registers and Vim has many registers. You can see contents present in all the Vim registers by running the Vim command `:reg`.

To yank a text to a particular register, you can prefix the yank command with `"{register}`. For example, to yank the text to register “1”, you can use the shortcut `"1yy`.

Similarly, you can paste the contents of a particular register. For example, `"2p` will paste the content present in register “2”.

Undo and Redo

`u`	Undo the last action.
`U`	Redo the last action.

These commands can be repeated by prefixing a number. For example, to undo last 3 actions, you can use `3u`.

Toggling Case

`~`	Toggle case at the current cursor position.
`gUU`	Make current line uppercase.
`guu`	Make current line lowercase.

You can also use `gu` and `gU` with a motion. For example, to convert 3 lines to uppercase, you can use `gU3j`.

Repeat Last Change

`.`	Repeats the last change.

This is where the real power of Vim comes in. For example, say you need to replace all occurrences of a word. You can first search for the word using `/` or `?`. Then to change the word you can use `ciw`. After changing go back to Normal mode, hit `n` to go to the next occurrence. Now you can simply press `.` to replace the word.

What's Next?

Once you get hold of these Vim keyboards shortcuts, open Vim and run the command `:help` to open the Vim documentation or you can use the online version of Vim documentation. It provides a list of every command there is with an explanation. So, you can pick up a few more useful Vim shortcuts and also get a better understanding.

Vim Commands Cheat Sheet

Source: Vim Commands Cheat Sheet .

Introduction

Vim is a widely used, open-source Unix text editor. Learning to use Vim commands is a matter of practice and experience. That is why it is handy to have a helpful reference sheet while mastering them.

In this tutorial, you will find the most important Vim commands as well as a downloadable cheat sheet.

Vim commands: Cheat Sheet.

Moving Inside a File

You can move the cursor within a file by single characters, words, tokens, or lines.

According to Vim, a word can be a group of letters, numbers, and underscores. On the other hand, a token is anything separated by whitespace and can include punctuation.

Additionally, you can move to different parts of a text by screen view.

Moving by Characters, Words and Tokens

The basic keys for moving the cursor by one character are:

- `h` – move the cursor left
- `j` – move the cursor down
- `k` – move the cursor up
- `l` – move the cursor right

You can also use these keys with a number as a prefix to move in a specified direction multiple times. For example, if you run `5j` the cursor moves down 5 lines.

- `b` – move to the start of a word
- `B` – move to the start of a token
- `w` – move to the start of the next word
- `W` – move to the start of the next token
- `e` – move to the end of a word
- `E` – move to the end of a token

For instance, you have the noun phrase “step-by-step” as part of a text and the cursor is placed at the end of it. The first time you press `b`, the cursor moves back to “step-by-**s**tep”. However, if you use `B`, the cursor moves all the way back to: “**s**tep-by-step” since there is no whitespace between these characters.

Moving by Lines

- `0` (zero) – jump to the beginning of the line
- `$` – jump to the end of the line
- `^` – jump to the first (non-blank) character of the line
- `#G` / `#gg` / `:#` – move to a specified line number (replace **#** with the line number)

To illustrate the difference between `0` and `^`, take a look at the following example. In the first bullet, the command moves the cursor to the blank space before the bullet. On the other hand, in the third bullet, the `^` key moves the cursor to the hyphen (the first character in the line).

Move the the beginning of line in Vim.

To learn more about `matchpairs` and how to use more than the default supported pairs, run the following commands in the text editor: `:h matchpairs`.

Commands for finding matchpairs in Vim.

Moving by Screens

The following commands are used as a quick way to move within the text without scrolling.

- `Ctrl + b` – move back one full screen
- `Ctrl + f` – move forward one full screen
- `Ctrl + d` – move forward 1/2 a screen
- `Ctrl + u` – move back 1/2 a screen
- `Ctrl + e` – move screen down one line (without moving the cursor)
- `Ctrl + y` – move screen up one line (without moving the cursor)
- `Ctrl + o` – move backward through the jump history
- `Ctrl + i` – move forward through the jump history
- `H` – move to the top of the screen (H=high)
- `M` – move to the middle of the screen (M=middle)
- `L` – move to the bottom of the screen (L=low)

Inserting Text

- `i` – switch to insert mode before the cursor
- `I` – insert text at the beginning of the line
- `a` – switch to insert mode after the cursor
- `A` – insert text at the end of the line
- `o` – open a new line below the current one
- `O` – open a new line above the current one
- `ea` – insert text at the end of the word
- `Esc` – exit insert mode; switch to command mode

Some of these commands switch between **command** and **insert mode**. By default, Vim launches in command mode, allowing you to move around and edit the file. To switch to command mode, use the **Esc** key.

On the other hand, the insert mode enables you to type and add text into the file. To move to insert mode, press `i`.

Switch to insert mode.

Editing Text

- `r` – replace a single character (and return to command mode)
- `cc` – replace an entire line (deletes the line and moves into insert mode)
- `C` / `c$` – replace from the cursor to the end of a line
- `cw` – replace from the cursor to the end of a word
- `s` – delete a character (and move into insert mode)
- `J` – merge the line below to the current one with a space in between them
- `gJ` – merge the line below to the current one with no space in between them
- `u` – undo
- `Ctrl` + `r` – redo
- `.` – repeat last command

Note: Bear in mind that Vim undoes and redoes changes by entries (changes made within one insert mode session). For more details, refer to the article How to Undo and Redo Changes in Vim.

Cutting, Copying And Pasting

- `yy` – copy (yank) entire line
- `#yy` – copy the specified number of lines
- `dd` – cut (delete) entire line
- `#dd` – cut the specified number of lines
- `p` – paste after the cursor
- `P` – paste before the cursor

Marking Text (Visual Mode)

Apart from command mode and insert mode, Vim also includes **visual mode**. This mode is mainly used for marking text.

Based on the chunk of text you want to select, you can choose between three versions of visual mode: **character mode**, **line mode**, and **block mode**.

- `v` – select text using character mode
- `V` – select lines using line mode
- `Ctrl` + `v` – select text using block mode

Once you have enabled one of the modes, use the navigation keys to select the desired text.

Versions of visual mode in Vim.

- `o` – move from one end of the selected text to the other
- `aw` – select a word
- `ab` – select a block with ()
- `aB` – select a block with {}
- `at` – select a block with <>
- `ib` – select inner block with ()
- `iB` – select inner block with {}
- `it` – select inner block with <>

Visual Commands

Once you have selected the desired text in visual mode, you can use one of the visual commands to manipulate it. Some of them include:

- `y` – yank (copy) the marked text
- `d` – delete (cut) the marked text
- `p` – paste the text after the cursor
- `u` – change the market text to lowercase
- `U` – change the market text to uppercase

Search in File

- `*` – jump to the next instance of the current word
- `#` – jump to previous instance of the current word
- `/pattern` – search forward for the specified pattern
- `?pattern` – search backward for the specified pattern
- `n` – repeat the search in the same direction
- `N` – repeat the search in the opposite direction

Saving and Exiting File

- `:w` – save the file
- `:wq` / `:x` / `ZZ` – save and close the file
- `:q` – quit

- `:q!` / `ZQ` – quit without saving changes
- `:w new_file_name` – save the file under a new name and continue editing the original
- `:sav` – save the file under a new name and continue editing the new copy
- `:w !sudo tee %` – write out the file using sudo and tee command

Working with Multiple Files

- `:e file_name` – open a file in a new buffer
- `:bn` – move to the next buffer
- `:bp` – go back to previous buffer
- `:bd` – close buffer
- `:b#` – move to the specified buffer (by number)
- `:b file_name` – move to a buffer (by name)
- `:ls` – list all open buffers

List all buffers in Vim.

- `:sp file_name` – open a file in a new buffer and split viewport horizontally
- `:vs file_name` – open a file in a new buffer and split viewport vertically
- `:vert ba` – edit all files as vertical viewports
- `:tab ba` – edit all buffers as tabs
- `gt` – move to next tab
- `gT` – move to previous tab

Open files as tabs in Vim.

- `Ctrl+ws` – split viewport
- `Ctrl+wv` – split viewport vertically
- `Ctrl+ww` – switch viewports
- `Ctrl+wq` – quit a viewport
- **Ctrl+wx** – exchange current viewport with next one
- `Ctrl+=` – make all viewports equal in height and width

Marks and Jumps

- `m[a-z]` – mark text using character mode (from `a` to `z`)
- `M[a-z]` – mark lines using line mode (from `a` to `z`)
- `` `a `` - jump to position marked `a`
- `` `y`a `` – yank text to position marked **>** `a>`
- `` `. `` – jump to last change in file
- `` `0 `` – jump to position where Vim was last exited
- ``` `` ``` – jump to last jump

- `:marks` – list all marks
- `:jumps` – list all jumps
- `:changes` – list all changes
- `Ctrl+i` – move to next instance in jump list
- `Ctrl+o` – move to previous instance in jump list
- `g,` – move to next instance in change list
- `g;` – move to previous instance in change list

Macros

- `qa` – record macro `a`
- `q` – stop recording macro
- `@a` – run macro `a`
- `@@` – run last macro again

Enabling Vim Color Schemes

- `:colorscheme [colorscheme_name]` – change to specified scheme
- `:colorscheme [space]+Ctrl+d` – list available Vim color scheme

The list of Vim color schemes shows you the ones that come by default with the text editor, as in the image below:

List Vim color schemes.

You can also configure the color settings manually or download user-made schemes. Find out how to do so in How to Change and Use Vim Color Schemes.

This article includes a one-page Vim commands reference sheet. Save the cheat sheet in PDF format by clicking the **Download Cheat Sheet** button below.

DOWNLOAD Cheat Sheet

Vim commands cheat sheet.

Conclusion

Knowing basic Vim commands is useful as most Linux distributions have it installed by default. Once you get use to using Vim commands, mastering Vim should be simple.

Until then, keep a Vim cheat sheet at hand.

Révision #3
Créé 17 juillet 2022 12:27:55 par Mickaël G.
Mis à jour 29 août 2023 19:49:23 par Mickaël G.