Cheat Sheet

R. Qudsi
University of Delaware, Newark, DE
Last updated on: 2021-05-29 @ 18:27

Contents

1 VIM

1.1 Global

:help keyword → open help for keyword
:o file → open file
:saveas file → save file as
:close → close current pane

1.2 Cursor Movement

h → move cursor left
j → move cursor right
k → move cursor up
l → move cursor down
H → move to top of screen
M → move to middle of screen
L → move to bottom of screen
w → jump forward to the start of a word
W → jump forward to the start of a word (words can contain punctuation)
e → jump forward to the end of a word
E → jump forward to the end of the word (words can contain punctuation)
b → jump backward to the start of a word
B → jump backward to the start of a word (words can contain punctuation)
0 → jump to the start of a line
^ → jump to first non-blank character of the line
$ → jump to the end of the line
g → jump to the last non-blank character of the line
gg → go to the first line of the document
G → go to the last line of the document
5G → go to line 5
:x → go to line number **x**
fx → jump to the next occurrence of character **x**
tx → jump to before next occurrence of character **x**
} → jump to next paragraph (or function block, when editing code)
- → jump to previous paragraph (or function block when editing code)
zz → center cursor on screen
Ctrl+b → move back one full screen
Ctrl+f → move forward one full screen
Ctrl+d → move down half screen
Ctrl+u → move up half screen

Tip: Prefix a cursor movement command with a number to repeat it. For example, 4j moves down 4 lines

1.3 Insert Mode

i → insert before the cursor
I → insert at the beginning of the line
a → insert (append) after the cursor
A → insert (append) at the end of the line
o → append (open) a new line below the current line
O → append (open) a new line above the current line
ea → insert (append) at the end of the word
Esc → exit insert mode

1.4 Editing

r → replace a single character
R → replace multiple character
J → join line below to the current line
cc → change (replace) entire line (enters the insert mode)
cw → change (replace) to the end of the word
c$ → change (replace) to the end of the line
s → delete character and substitute text
S → delete line and substitute text (same as cc)
xp → transpose two letters (delete and paste)
u → undo
Ctrl+r → redo
. → repeat last command

1.5 Visual Mode

v → start visual mode, mark line, then do a command (like y-yank)
V → start linewise visual mode
Ctrl+v → Start blockwise visual mode
o → move to the end of marked area
O → move to the other corner of block
xaw → mark **x** words
ab → a block with ()
aB → a block with {}
ib → inner block with ()
iB → inner block with {}
Esc → exit visual mode
≥ → shift text right
≤ → shift text left
y → yank (copy) marked text
d → delete marked text
~ → switch case

1.6 Cut and Paste

yy → yank (copy) a line
2yy → yank (copy) 2 lines
yw → yank (copy) the characters of the word from the cursor position to the start of the next word
y$ → yank (copy) to end of line
p → put (paste) the clipboard after cursor
P → put (paste) before cursor
dd → delete (cut) a line
xdd → delete (cut) **x** lines

xdw → delete (cut) **x** words from the cursor position to the start of the next word
D → delete (cut) to the end of the line
d$ → delete (cut) to the end of the line
d^ → delete (cut) to the first non-blank character of the line
d0 → delete (cut) to the beginning of the line
x → delete (cut) character

1.7 Exiting

:w → write (save) the file, but don't exit
:w !sudo tee % → write out the current file using sudo
:wq or :x or ZZ → write (save) and quit
:q → quit (fails if there are unsaved changes)
:q! or ZQ → quit and throw away unsaved changes

1.8 Search and Replace

1.8.1 Search

/pattern → search for **pattern**
?pattern → search backward for **pattern**
vpattern → 'very magic' pattern: non-alphanumeric characters are interpreted as special regex symbols (no escaping needed)
:%s/pattern//gn → count the number of matches of a **pattern** in the current buffer
:%s/pattern//n → count the number of matches of a **pattern**
:a,bs/pattern//gn → count the number of matches of a **pattern** from line **a** to **b**
:'<,'>s/pattern//gn → count the number of matches of a **pattern** in the lines in the most recent visual selection
:%s///gn → count the number of occurrences of the last used search pattern
n → repeat search in same direction
N → repeat search in opposite direction
:noh → remove highlighting of search matches

1.8.2 Replace

:%s/old/new/g → replace all **old** with **new** throughout file
:%s/old/new/gc → replace all **old** with **new** throughout file with confirmations
:%s/old/new/gci → replace all **old** (case sensitive because of flag **i**) with **new** throughout file with confirmations (%**s/old\c/new/gci** does the same thing)
:%s/\ ⟨*old*\⟩ /new/gc → replace all **old** (exact pattern) with **new** throughout file with confirmations
:.,+as/foo/bar/g → replace all **old** with **new** for the current line (.) and the **a** lines.
:g/^word/s/old/new/g → replace all **old** with **new** for each line that starts with **word**
:s/foo/bar/g → replace all **old** with **new** in the current line.
:A,Bs/foo/bar/g → replace all **old** with **new** from line **A** to **B**.
Notes: /\t is tab, /\s is white space, /\n is new line, in order to replace a line though use \r since \n just introduces a null character
After an opening [, everything until the next closing] specifies a collection.

1.9 Sessions

:mksession ~/x.vim → create a session named **x.vim** and save it
:source ~/x.vim → restore session named **x.vim**, while in vi
$ vim -S ~/x.vim → restore a session named **x.vim** (in command line)

1.10 Miscellaneous

:ab word1 word2 → change word1 to word 2 (helps to store abbreviations or avoid common typos)
:una word → remove word from the list of abbreviations
:ab → lists all the abbreviations being used

Tip: The abbreviations work at local level, and once the session gets over those are lost. In order to make it universal, go to '.vimrc' and add the command ***:iabbrev w1 w2***

1.11 Marks

:marks → list of marks
ma → set current position for mark A
'a → jump to position of mark A
y'a → yank text to position of mark A

1.12 Multiple Tabs and windows

vi -p x y z → open files **x, y** and **z**
:tabnew x → open file **x** in a new tab
:tabc → close the tab
:tabn → switch to next tab
:tabp → switch to previous tab
:tabn x → move to $\mathbf{x}^{th}$ tab
gt or :tabnext or :tabn → move to the next tab
gT or :tabprev or :tabp → move to the previous tab
xgt → go to $\mathbf{x}^{th}$ tab
:tabr → move to first tab
:tabl → move to last tab
:tabs → list all open tabs
:tabmove x → move current tab to the $\mathbf{x}^{th}$ position (indexed from 0)
:tabclose or :tabc → close the current tab and all its windows
:tabonly or :tabo → close all tabs except for the current one
:tabdo command → run the command on all tabs (e.g. :tabdo q - closes all opened tabs)
:qa → close all open tabs
:wa → save all the open tabs
:qwa or :xa → save and exit all open tabs
:e x → open file **x** in this window (closes the previous file)
:ls → show the list of currently open buffers (windows)
:b x → go to buffer (window) **x**
:split x → split window horizontally and load file **x**
:vs x → split window vertically and open file **x** (readmode only)
:vsplit x → split window vertically and open file **x**
:hide → close the current window
:only → keep only the current window open
Ctrl + ww → switch windows
Ctrl + wq → quit a window
Ctrl + wv → split window vertically
Ctrl + wh/wl → move cursor to the left/right window (vertical split)
Ctrl + wj/wk → move cursor to the window below/above (horizontal split)
Ctrl + wT → move the current split window into its own tab
:e file → edit a file in a new buffer
:bnext or :bn → go to the next buffer
:bprev or :bp → go to the previous buffer
:bd → delete a buffer (close a file)
:ls → list all open buffers
:sp file → open a file in a new buffer and split window
:vsp file → open a file in a new buffer and vertically split window

1.13 Search in Multiple Files

:vimgrep /pattern/ file → search for pattern in multiple files
:cn → jump to the next match
:cp → jump to the previous match
:copen → open a window containing the list of matches

1.14 Registers

:reg → show registers content
"xy → yank into register **x**
"xp → paste into register **x**

Tip: Registers are stored in /.viminfo, and will be loaded again on next restart of vim.
Tip: Register 0 always contains the value of the last yank

1.15 Macros

qa → record macro **a**
q → stop recording macro
@a → run macro **a**
@@ → rerun last macro

1.16 Opening a file from another folder while being inside VIM

:Ex → Opens the pwd. Can navigate through the folders by moving up and down to the name of the folder and pressing Enter.
:Ex ⟨*directory*⟩ → Takes you to the **directory**. Once inside the directory, navigate to the file you want to open and press enter.

2 Terminal

source .bashrc → Updates terminal with the updated **.bashrc** file
history x → Displays last **x** used commands
history | grep str → Displays all commands which started with **str** (*You have aliased this command to* **ch** *(in .bashrc), so can just use* `ch str` *instead of history | grep str*).

ctrl+u → Deletes everything before cursor
ctrl+k → Deletes everything after cursor
ctrl+l → Clear the screen
nautilus /path/folder → Open the **folder**
!# → run command number #
!str → execute last command that began with **str**
!?str? → execute last command that contains **str** (and not necessarily starts with)
cat or less ~ /.bash_history → print out the history file
ls -1 | wc -l → counts the number of files in the present directory
du -command file/folder → has several uses based on the command given. Following are few examples:

- → Without any additional command, gives the size of each folder and sub-folders.
- -a → Gives the size of every folder and file
- -h → Outputs size in human readable format.
- -s → Gives the summary of sizes.
- -k/m → Gives size in kilobyte/ megabytes.
- -c → Gives total disk space in the last line.
- –exclude= '*.fmt' → Excludes displaying results of file with format fmt .
- –time → Shows disk usage based on last modification time and displays time of modification as well.

man command → Shows the manual for command
find command → Has several uses. Following are a few example:

- . → Display all the files and directory inside the present directory (replace dot with a directory name to display everything in that directory)
- . -type d/f → Display all the directory/file in the present directory
- . -type f -fname "test*" → Display the location of all the files and the name of the files which start with name **test** (use **-iname** instead of **-name** to make search case insensitive.)
- . -type f -mmin -/+t → Display all the files modified in less/more than **t** minutes (use **-mtime** for days)
- . -size +5M → Display all the files larger than **5 MB** in size (use 'k' for kilobytes and 'G' for gigabytes)
- . - empty → Display all the empty files

grep -winlr -A/B x "str" file.ext → Search for text **str** in the file **file.ext**.

- w → Only display the ones with the whole match
- i → Make the search case insensitive
- n → Display the line number
- l → Display only the files which has **str** (doesn't show line numbers) (using 'c' instead of 'l' which also display the number of matches in each file)
- r → Does recursive search, in the present and all the other sub-directories
- A/B → Display **x** number of lines after/before the place where **str** has been used
- C → Display **x**/2 number of lines before and after the place where **str** has been used

3 crontab

crontab -e → Gives you access (edit) to the crontab file.
In the edit mode type the task you want to schedule in the following way:
* * * * * command

- * means minute (0-59)
- * means hour (0-23)
- * means day of the month (1-31)
- * means month of the year (1-12)
- * means day of the week (0-6, Sunday to Saturday)

crontab -r → Removes any stored crontask!
Be very careful while executing this command. Will wipe out every stored tasks!

4 RSYNC

Notes 2: Use **crontab** to start the rsync options
rsync /dir/dir1/* dir2/ → Copies every file in /dir1/ into /dir2 (Set * to filename if you want to transfer a specific file)
rsync -r /dir/dir1/ dir2/ → Copies everything (including directories) from /dir1/ into /dir2
Notes 3: Use of '/' after dir1 is important. It makes sure that the content of dir1 is copied into dir2 and not the dir1 itself .

rsync -av -in (or - -dry-run) - -delete /dir/dir1/* dir2/ → Syncs files from /dir1 to /dir2 with various options.
i) -a : Stands for 'archive', recurses into directory like '-r' and preserves the information like modified date, owners etc.
ii) -av : 'v' stands for verbose, thus it prints out the list of files it is changing.
iii) -in (or - -dry-run) : Shows the list of files and folders which will will be copied (needs '-av' before it prints).
iv) - -delete : Completely syncs /dir1 to dir2, meaning, it will delete any file/folder from /dir2 which weren't present in /dir1. Be super careful while using this command.

5 VS Code

Note 3: For detailed descriptions click on the following link
Notes 4: By default, the **VS code** is in **vim** mode, so any short-cut that is also in **vim** won't work.
Ctrl+ or - → Change the font size of various UI elements
Ctrl+\→ Open another editor side by side (similar to **:vs** functionality of vim)
Shift+Alt → Box selection or column aligned selection
Alt → Fast scrolling (5x)
Alt+Up or Alt+Down → Move a selection of lines up or down (works in VI mode)
Shift+Alt+Left or Shift+Alt+Right → Shrink or expand selection
Shift+Alt+Up or Shift+Alt+Down → Copy the line below or above the present line
Ctrl+Shift+[or Ctrl+Shift+]→ Code folding or expanding (if folded)
Alt+F12 → Peek at the definition and options (a new small window opens up with definition, press Esc to close the new window)
F12 → Go to the definition file (opens a new tab)
select a word + F2 → Rename all occurrences of the selected word by renaming it to whatever.

6 GIT

git config –global credential.helper "cache –timeout=360" → Saves your credentials for **360 seconds** so that you won't have to enter password for every push and pull requests.
git add –a → add all the file for committing
git commit -a → Commit all the added files for pushing to the repository
git push → Push all the committed file to the online repository/branch
git config –global alias.hist "log –pretty=format: '%h %ad — %s%d [%an]' –graph –date=short" → Aliasing **hist** command for GitHub
git hist → Check the history of commits on GIT
git clone "url" "where to clone" → Clone a remote repository to your local directory
git merge "branch" → Merges the **branch** to the present branch (the one you are in)
git branch -d "branch" → Delete **branch** locally
git push origin –delete "branch" → Delete **branch** from the online repository
git checkout afe52 → checkout the commit based on the hash
git checkout 'master@1918-05-11 12:00:00' → Checkout based on date
git checkout @314.days.ago → Checkout based on day
git log → To check the commits
git commit -m "insert message here"→ Make commit along with a message
git log → To check the commits
git reset → Uncommit all the files
git reset –hard **ID** → will make local code and local history be just like it was at that commit
git reset –soft **ID** → will make local files changed to be like they were then, but leave your history etc. the same.
git cherry-pick **ID** → Sets the present repository to the state of other repository whose **ID** was used.
git stash save "message"→ Stash the changes and return the files to before the all the changes made to it
git stash list→ Lists out all the stashed instances along with their IDs
git stash apply "stash ID" → Apply the stashed file. (this way the Stash ID is still stored and doesn't get deleted)
git stash pop → Grabs the stash on the top of the list, applies it, and drop the stash
git stash drop ID → Drops the stash of that **ID**

Terminal	CMD	Description
ls -l	dir	Directory listing
mv	ren	Rename a file
cp	copy	Copy a file
mv	move	Move a file
clear/ctrl+l	cls	Clear screen
rm	del	Delete a file
rm -rf foo	rd /s /q foo	Delete the folder foo
diff	fc	Compare the file content
cd foo	chdir or cd foo	Change directory to foo
pwd	chdir	Show the current directory
mkdir	md or mkdir	Create a new directory
ls -R	tree	To list directory recursively
foo –help	foo /?	Shows manual for command foo
	tasklist	List all the tasks
	taskkill /IM foo.exe /F	Kill the tasks foo.exe
	taskkill /PID abcd /F	Kill the tasks with PID abcd
	color ab	Change color of background and foreground. a for background and b for foreground
	doskey/history	Display a history of previous commands
	F8	Goes through the past list of commands
	F9	Prompts you to input the command number for among the list of past commands
	foo — clip	Send output to clipboard
	sfc/scannow	scans Windows system files and looks for problems. If some files are missing or corrupted, this command fixes them

7 Active Aliases in .bash _aliases file

- ..="cd .."
- ...="cd ../.."
- ....="cd ../../.."
- .....="cd ../../../ .."
- ~ ="cd ~"
- dr="cd ~/Dropbox/Studies/Research"
- dl="cd ~ /Downloads"
- d="cd ~ /Desktop"
- gh="cd ~ /Desktop/GIT"
- fm="cd ~ /Desktop/GIT/ fm_development"
- pr="cd ~ /Desktop/GIT/Personal/Janus"
- dt="cd /data/Research/"
- dfm="cd /data/Research/Active_Research/fc /Janus"
- sjfm="vi -S ~ /session_janus_fm.vim"
- sjma="vi -S ~ /session_janus_master.vim"
- sjlo="vi -S ~ /session_janus_fm_local.vim"
- sjpr="vi -S ~ /session_janus_personal.vim"
- smms="vi -S ~ /session_mms.vim"
- ipfm="ipython –matplotlib =qt4 –profile = fm"
- ipma="ipython –matplotlib =qt4 profile = master"
- ippr="ipython –matplotlib =qt4 –profile = personal"
- ipar="ipython –matplotlib =qt4 –profile = active"
- week="date +%V" (shows week number)
- khamosh="something" (mutes the computer)
- bajao="something" (sets volume to maximum)