A Simple Collection of Cheat Sheets

Coding Club
SNU Chennai

Contents

Chapter 1

Linux Cheat Sheet

1.1 The *very very useful ones*

1.1.1 `man`

- `man <command name>`
- The main command returns a helpful help page that gives you a brief description of what exactly a command does and how to use it.
- In case that man doesn't work, make sure that you have `mandocs` or `man` installed.
- The manual entries can also be accessed via a browser.

1.1.2 `--help` or `-h` flag

- Sometimes a command is too niche to warrant a page in the manual. In such cases you can use the help flags. There is no set standard but they're usually one of the two
- `<command> --help` or `<command> -h`
- If one doesn't work, try the other.

1.1.3 `sudo` and `su`

- `su` opens the current shell as root while `sudo` runs a specified command as root.
- In Linux root privileges are very similar to Administrator privileges in Windows
- Be very careful when running anything as root. It can break your system.--
- It also requires the current user to know the root password as well be a part of the sudoers list

1.2 The Essentials

1.2.1 `cd`

- `cd` is used to traverse the file system from the terminal
- `cd <path>` will move the terminal to the defined path
- `cd ..` will move the terminal to the parent directory if it exists
- `cd /` will move the terminal to the root directory
- `cd ~` or just `cd` will move the terminal to the home directory

1.2.2 `ls`

- `ls` shows all the visible files and folders in directory.

- `ls -a` shows all the hidden and visible files and folders in the directory.
- `ls -A` shows **almost** all the hidden and visible files and folders in the directory(It excludes the `.` which loops back to the current directory and the `..` which points to the parent directory)
- `ls -l` displays all the visible files and folders and lists them in tabular form with some extra information(like size, author, date modified, etc)

1.2.3 `touch`

- `touch <filename>` creates an empty folder with the name specified

1.2.4 `mkdir`

- `mkdir <foldername>` creates a folder with the specified name in the current location of the terminal.
- `mkdir -p <path_to_folder>` creates the folder in the path specified as well as all the missing folders in the path to the folder.

1.2.5 `mv`

- `mv <source_path> <destination_path>` moves a file from the source_path to the destination_path
- `mv <path>/old_name <path>/new_name` will rename a file named `old_name` in the location `<path>` to `new_name`

1.2.6 `cp`

- `cp <source_path> <destination_path>` copies a file from the source_path to the destination_path
- `cp -r` is used to copy folders by recursively copying their contents

1.2.7 `rm`

- `rm <file1> <file2> <file3>` deletes the files mentioned
- **The Recycle Bin or Trash does not exist when it comes to deleting things from the terminal** so be careful with this command.
- `rm -f <file>` force deletes the file and overrides any warnings.
- `rm -r <folder>` recursively deletes the contents of a folder and finally deletes the folder too.
- **UNDER NO CIRCUMSTANCE SHOULD YOU RUN ANY OF THE FOLLOWING UNLESS YOU ARE ABSOLUTELY SURE ABOUT WHAT YOU ARE DOING**
  - `sudo rm -rf /` this will delete everything in your root directory
  - `rm -rf ~` this will delete everything in your home directory
- Running `rm -rf` with elevated privileges in a dangerous location will most probably break your OS.
- Exercise caution.

1.2.8 `grep`

- `grep` is used to search the content of a specified file or directory for a given string or a regex
- `grep '<search_term>' <file>` searches for the `search_term` in the `<file>`
- `grep -i '<search_term>' <file>` will search for the `search_term` in a case insensitive way.
- `grep -r '<search_term>' <folder_path>` will search for the `search_term` recursively within the specified directory.

1.2.9 `cat`

- `cat` allows the user to execute basic text modification from the terminal
- It is not a full blown editor like Vim or Emacs but it can read and append to files
- `cat <file>` will display the contents of the file
- `cat >> <file>` will allow you to enter some text into the terminal. The entered text is then appended to the end of the file.

1.3 The Extras

1.3.1 `pwd`

- pwd returns the path to the active directory

1.3.2 `top`

- `top` displays the processes running in real time. It also displays resource utilisation and other information regarding the process.

1.3.3 `pkill`

- `pkill <pattern>` kills the first processes with the string `<pattern>` in their name

1.3.4 `pgrep`

- `pgrep <pattern>` returns the PID of all processes with the string `<pattern>` in their name

1.3.5 `kill`

- `kill <PID>` terminates the process with the PID specified

1.4 Useful for Bash Scripting

1.4.1 `echo`

- `echo "Hello, World!"` will print Hello, World! on the terminal.

1.4.2 `wc`

- `wc <file>` will print the number of words in a file
- `wc -l <file>` will print the number of lines in a file

1.4.3 Piping data

- We can pass the output of one command to the input of another by using the `|` operator.
- `grep -r '<search_term>' <folder_name> | wc -l` will return the number of entries that match the search

Chapter 2

Git Cheat Sheet

2.1 The most basic and essential commands.

2.1.1 Installation

After you have downloaded git from the URL you can continue with the content below.

2.1.2 Setup

`git config --global user.email "yourEmail"`

- This commands let's you set up a email id that will be displayed in your commits

`git config --global user.name "yourName"`

- This commands let's you set up a name that will be displayed in your commits

2.1.3 Initialisation

`git init`

- This initialises a new git repository locally in your project root
- Follow the commands below to start using version control software

`git add filename.filetype`

- Adds the mentioned file to the staging area
- Just replace the filename.filetype to your file name
- To add everything in your repository just use `git add .`

`git status`

- To show the current status of your repository
- The status includes staged,unstaged and untracked files.

`git commit`

- This will open up an editor to add the commit message to the changes done in the repositories.
- If you don't want a editor to open up every time you make a commit use the command `git commit -m "your message"`

`git log`

- This will show the commit history for the repository along with the commit id for each commit.
- you may even use `git log -p` to view the commit history along with the all the files and their changes.

2.1.4 How to ignore files in git

- Create a `.gitignore` file and commit it
- Add all the files that you want to ignore in your git repository inside the `.gitignore` file

2.1.5 Roll back to a older or a specific version

`git revert commit_ID`

2.1.6 Branches in git

`git branch branchName`

- By default you will be in the main branch. With this command you can create new branch
- Git wont automatically switch to any branch you should change branch with the next command

`git checkout branchName`

- Used to switch to a specific branch
- To get a list of existing branches use the command `git branch`

`git merge brancName`

- Incorporates the changes of another branch to the current or main branch.

2.1.7 How to add and work on a remote repository in Git

`git remote add origin https://repo_here`

- Adds your local repository to a remote repository (on GitHub, GitLab or any other cloud based git hosting domains).

`git clone https://remote_repo_url`

- Creates a local copy of an existing remote repository.

`git push`

- After you are done working your local repository you can push the changes into your remote repository using the above command.

`git pull`

- This will merge the changes of your remote repository with your local repository.
- Remember to commit the changes of your local repository before using `git pull`.