


bash and git commands

M. Massias

1 bash

These commands are for Linux/OS X; if you use the Windows shell `cmd.exe`, search for their equivalent names in Windows. Instead of using that shell, we recommend to use the program “Git for Windows”. Remember to use autocompletion with  to go faster.

Names between angles brackets (< and >) are supposed to be replaced by the actual names of the files/folders considered.

1. `pwd`: print working directory (where am I in my filesystem?)
2. `cd <address>`: go to this address (stands for “change directory”). Can be chained, separating folders by slashes: `cd <folder1>/<folder2>`.
`cd ..` goes to the parent folder, `cd ../../` goes two folders up, etc.
`cd` alone takes you to your home directory, also denoted by `~`.
3. `ls`: list all files and folder in current directory. Useful option: `ls -la` (list everything including hidden files, and present the results as a list). It is the same as `ls -l -a`.
4. `man ls` get help (manual) for a command, here for `ls`. `ls --help` or its shortcut `ls -h` usually work for most commands (beyond `ls`), except on OSX.
5. `mv <existing_file_or_folder> <new name>`: move, or rename. Example: `mv <file1> <file2>` renames the (existing) `file1` to a new file, `file2`.
`mv <file1> <dir1>` moves `file1` inside the (existing) `dir1` directory.
6. `mkdir <directory name>`: create a directory (stands for “make directory”)
7. `cp <file1> <file2>`: copy the 1st file into the second
8. `cat <filename>`: print the content of a file, which can be quite long! Stands for “concatenate”.
9. `head -n 10 <filename>`: print the 10 first lines of a file (useful for large files)
10. `rm <filename>`: remove file.
11. `rm -r <directory>`: remove recursively, ie remove the directory and everything inside.

12. **which <program>**: tells you which executable is actually called when you call **program**. Useful cases: **which python** to check which interpreter is being called (your base one, or the one from an environment, or another one on your system), also see **which conda** when you may have multiple installs (e.g. both miniconda and conda).
13. **touch <filename>**: create and empty file.
14. **echo <text>**: equivalent to print in python. **echo <text> > <filename>** directly writes the text to the target file (and overwrites the contents of said file).
15. the symbol **|** is called a pipe (**alt** + **L** on Mac) and used to chain commands together. Example: **cat <filename> | grep <word>** first outputs the contents of the file, then selects only the lines containing the word.
16. **wc <filename>**: counts the words in the file. **wc -l <filename>** counts the number of lines.
17. **clear**: clear your terminal.

You can use the up and down arrows to move up and down in the history of commands that you have typed (i.e. pressing **↑** once puts the last command that you typed, etc.)

1.1 Basic text editors

You should learn how to use basic functions of a text editor which does not involves a Graphical User Interface (no buttons to click). We recommend using nano. To quit nano: **ctrl** + **O**, validate the name of the file with **↵**, then **ctrl** + **X**. These shortcuts are recalled on the 2 last lines when you open nano.

To start writing in vim: press **i** (activate insert mode). To quit vim: press **Esc**, then type **:q!** (quit without writing, adding a bang forces the exit) or **:wq** (write and quit), and press **↵**.

1.2 More advanced

Linux and Mac users may want to install **zsh**, a more user friendly version of bash (terminal): <https://gist.github.com/derhuerst/12a1558a4b408b3b2b6e>.

If you do so, it is recommended to add the following 2 lines at the end of the file **.zshrc** (which is located in your home directory):

```
source ~/.bash_rc
source ~/.bash_profile
```

Amongst other perks, zsh makes it way easier to navigate through command history with **↑** and **↓**. If you use conda, you must do **conda init zsh**, to configure conda with zsh.

2 git

1. `git clone <address>` where `address` is copied from github when you hit the “clone” button. when you clone, the repo is downloaded in your current working directory. You must then use `cd` to enter it.
2. `git config --global user.name "YourName"`: globally sets your user-name (to do once only, when you start using git on a new machine);
- `--global` is an option, telling git to set this configuration for every git project on your computer. You can put any name as user name, it does have to match your github name. It's the name which will appear in the commit list

You should also configure your email with
`git config --global user.email "yourmail@smthng.fr"`.
3. `git config`: show all config variables. Combine it with `grep` to get the value on one particular option.
4. `git log -4`: show the last 4 commits on your computer. Use `enter` to navigate the editor that this commands opens (called `less`¹), and type `q` if you want to exit it.
5. `git add <filename>`: start tracking a new file, or stage a currently tracked (and modified since last commit) file to be committed in the next commit.
6. `git commit -m "Your_commit_title"` does a commit including all currently staged files.
7. `git push`: publish your commit on the remote repository.
8. `git pull`: get the latest commits from the remote repository.
9. `git remote -v`: see the address of your remote repository (potentially, repositories).
10. `git fetch`, `git rebase`: similar ideas to pull and merge, but done in a different fashion.²
11. `git reset --hard HEAD^`: deletes last commit (and its modifications). Do not do it if you have already pushed your alst commit, as you won't be able to push in a clean fashion. If you want to undo an already published commit, use:
12. `git revert <your_commit_hash>`, where `<your_commit_hash>` is a number like `444b1cff` (get it via the `git log` command mentioned above)

¹<https://stackoverflow.com/questions/9483757/how-to-exit-git-log-or-git-diff>

²<https://stackoverflow.com/questions/16666089/whats-the-difference-between-git-merge-and-git-rebase>