Introduction to HPC@UoP

Vincent Drach

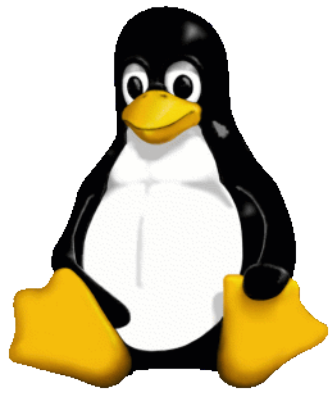
School of Computing, Electronics and Mathematics
vincent.drach@plymouth.ac.uk

Linux & HPC environment

UNIVERSITY OF PLYMOUTH

Today’s goals

- Quick tour of the Linux operating system (OS)
- Access foseres and configure your working environment
- Submit a job on 1 computing node


Tux is the official mascot of the Linux kernel.

Outline

- Introduction :
  - ★ High performance computing
  - ★ Operating system: Linux
- Linux Tutorial (using Ubuntu):
  - ★ Linux Filesystem
  - ★ Basic Unix tools
  - ★ Scripting
- Specificities of an HPC environment
  - ★ Remote connection
  - ★ Modules
  - ★ Queuing system
  - ★ Virtual environment

Introduction

High Performance Computing

- The aim of High Performance Computing is to use a large number of computing resources to solve computation problems efficiently.

- Computing resources:
  - Multi-core CPU
  - GPUs
  - FPGA

- A cluster combines the computing nodes through a network that allows to communicate data.

- Depending on your scientific problem
  - You use a software that do not require to use multiple nodes
  - You use a software that orchestrate the communication between the nodes for you
  - Some adjustment to design a parallelisation strategy is required.
  - Some significant development are required

- High Performance Computing environment requires
  - To analyse the computing problem that you are trying to solve
  - To accept that part of the research is to learn how to use the hardware
  - To do some experimentation

Program: definition

- A **computer program** is a collection of instructions[1] that performs a specific task when executed by a computer. A computer requires programs to function and typically executes the program's instructions in a central processing unit.

- the program in its human-readable form is called source code

- Two (main) categories:

  ★ Source code can be converted by a compiler to derive machine code—a form consisting of instructions that the computer can directly execute usually referred to as executables

  ★ Alternatively, a source code may be executed with the aid of an interpreter:, a program that executes source code from a programming language line-by-line.

- An algorithm is a well-defined sequence of operation and is not to be confused with its implementation

The algorithm is the cooking recipe - the program is the cake !

What is an Operating System?

- Operating system (OS) = Software that sits between applications and hardware
  - ★ Has privileged access to hardware
  - ★ Provides services and interfaces to applications

- User applications call OS routines for access and services
- OS contains computer programs, device drivers, and the kernel
- The **kernel** is the central part of an operating system. It manages the operations of the computer and the hardware - most notably memory and CPU time

What is the role of the OS?

- Provides a layer of abstraction for hardware resources
    - ★ Allows user programs to deal with higher-level, simpler and more portable concepts
    - ★ Hide underlying details, and provide cleaner, easier-to-use, more elegant concepts and interfaces
    - ★ Also provides standardised interfaces despite diversity of implementation underneath
- Manages the ressources
    - ★ Allow multiple applications to share resources without hurting one another
    - ★ Allow multiple users to share resources without hurting one another
    - ★ OS dynamically manages which applications get how many resources
- Protects programs and their data from one another, as well as users from one another
    - ★ what if I could modify your data, either on disk or while your program was running?

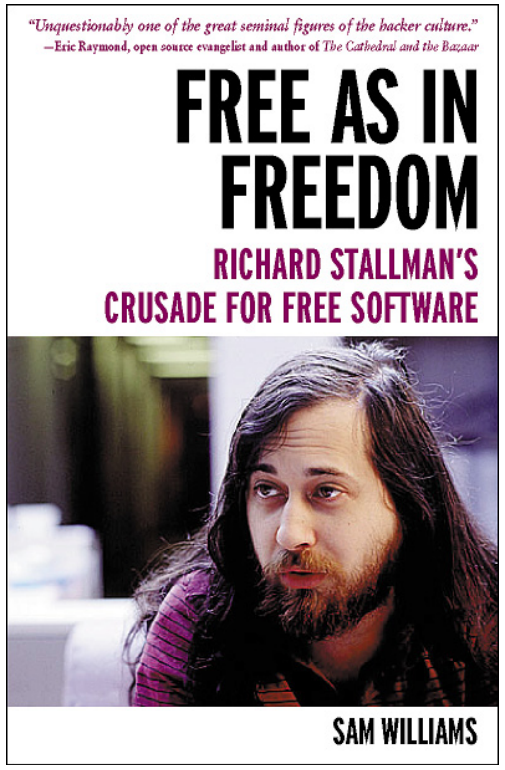
Example of “abstraction”: hard disk

- Disk hardware and operations are very complex
  - ★ Multiple heads, cylinders, sectors, segments
  - ★ Have to wait for physical movement before writing or reading data to/from disk
  - ★ Data stored discontinuously for performance
  - ★ Sizes and speeds are different on different computers
- OS provides simple read() and write() calls as the application programmer’s interface
  - ★ Manages the complexity transparently, in conjunction with the disk controller hardware

Before Linux

- main OS in the 80’s:
  - ★ Microsoft’s DOS
  - ★ Apple MAC
  - ★ UNIX
- Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected (No modification is possible without paying high license fees)

- 1984: GNU project
  - ★ Development of Started by Richard Stallman who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
  - ★ GNU is a recursive acronym for "*GNU's Not Unix!*”, chosen because GNU's design is Unix-like
  - ★ Stallman built the first free GNU C Compiler in 1991.


“Unquestionably one of the great seminal figures of the hacker culture.”
—Eric Raymond, open source evangelist and author of *The Cathedral and the Bazaar*

FREE AS IN FREEDOM

RICHARD STALLMAN'S CRUSADE FOR FREE SOFTWARE

SAM WILLIAMS

Beginning of Linux

- In Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1

- Licensed under GNU General Public License,thus ensuring that the source codes will be free for all to copy, study and to change.

Linux nowadays

- Linux was originally developed for personal computers based on the Intel x86 architecture, but has since been ported to more platforms than any other operating system

- Because of the dominance of the Linux kernel-based Android OS on smartphones, Linux has the largest installed base of all general-purpose operating systems.[

- All the world's 500 most powerful supercomputers use a linux based operating system.

- The development of Linux is one of the most prominent examples of free and open-source software collaboration. The underlying source code may be used, modified and distributed—commercially or non-commercially—by anyone under the terms of its respective licenses

- The most popular and mainstream Linux distributions are Arch Linux, CentOS, Debian, Fedora, Gentoo Linux, Linux Mint, Mageia, openSUSE and Ubuntu, together with commercial distributions such as Red Hat Enterprise Linux and SUSE Linux Enterprise Server

- A **Linux distribution** (often abbreviated as **distro**) is an operating system made from a software collection, which is based upon the Linux kernel and, often, a package management system.

Linux tutorial

Login in SMB306

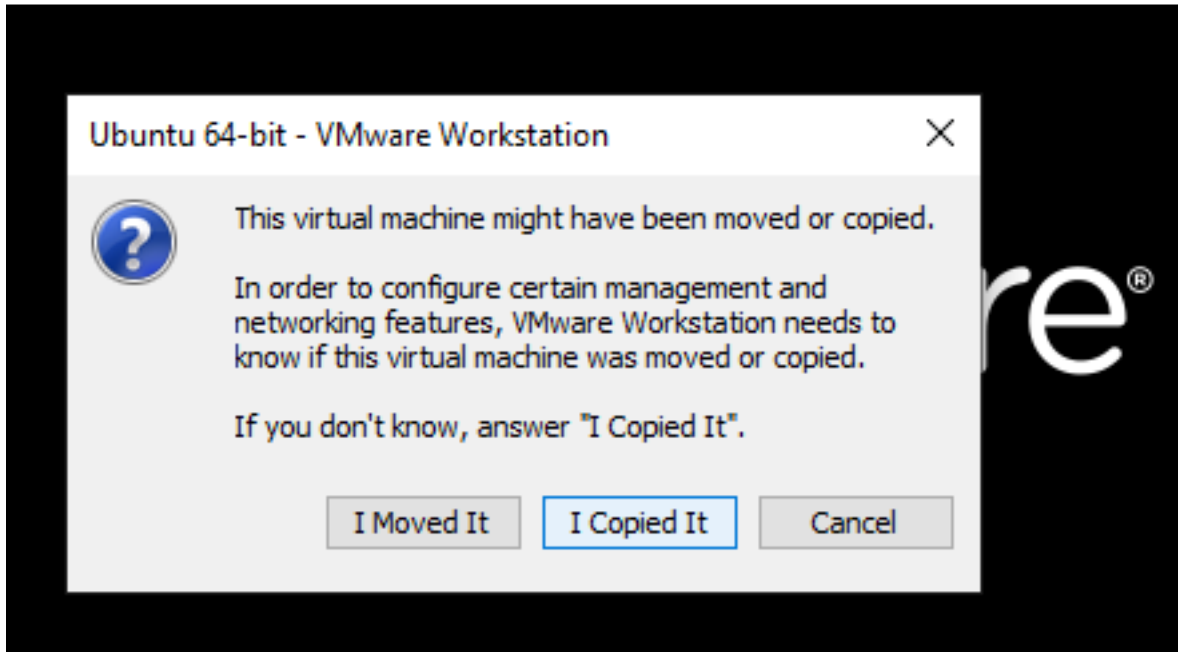
- Login: HPC-user
- Password: Training22

- This is a shared account (do not store any password)

- If you want to back up things, remember to upload them to your one drive.

- Start the Ubuntu virtual machine (VM)

- Then click on Power on this virtual machine
- Then click on "I copied It"



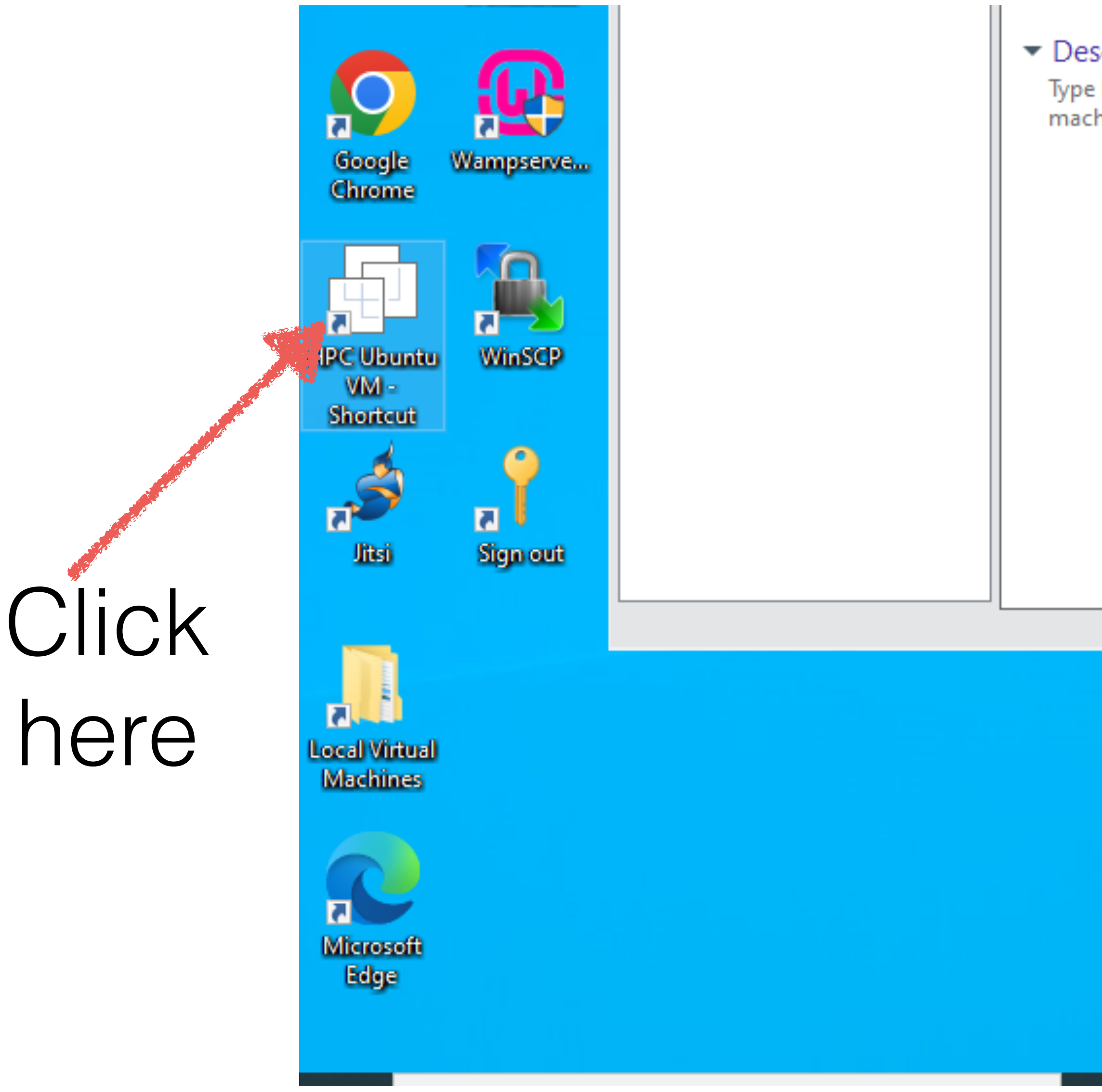
Ubuntu 64-bit - VMware Workstation

This virtual machine might have been moved or copied.

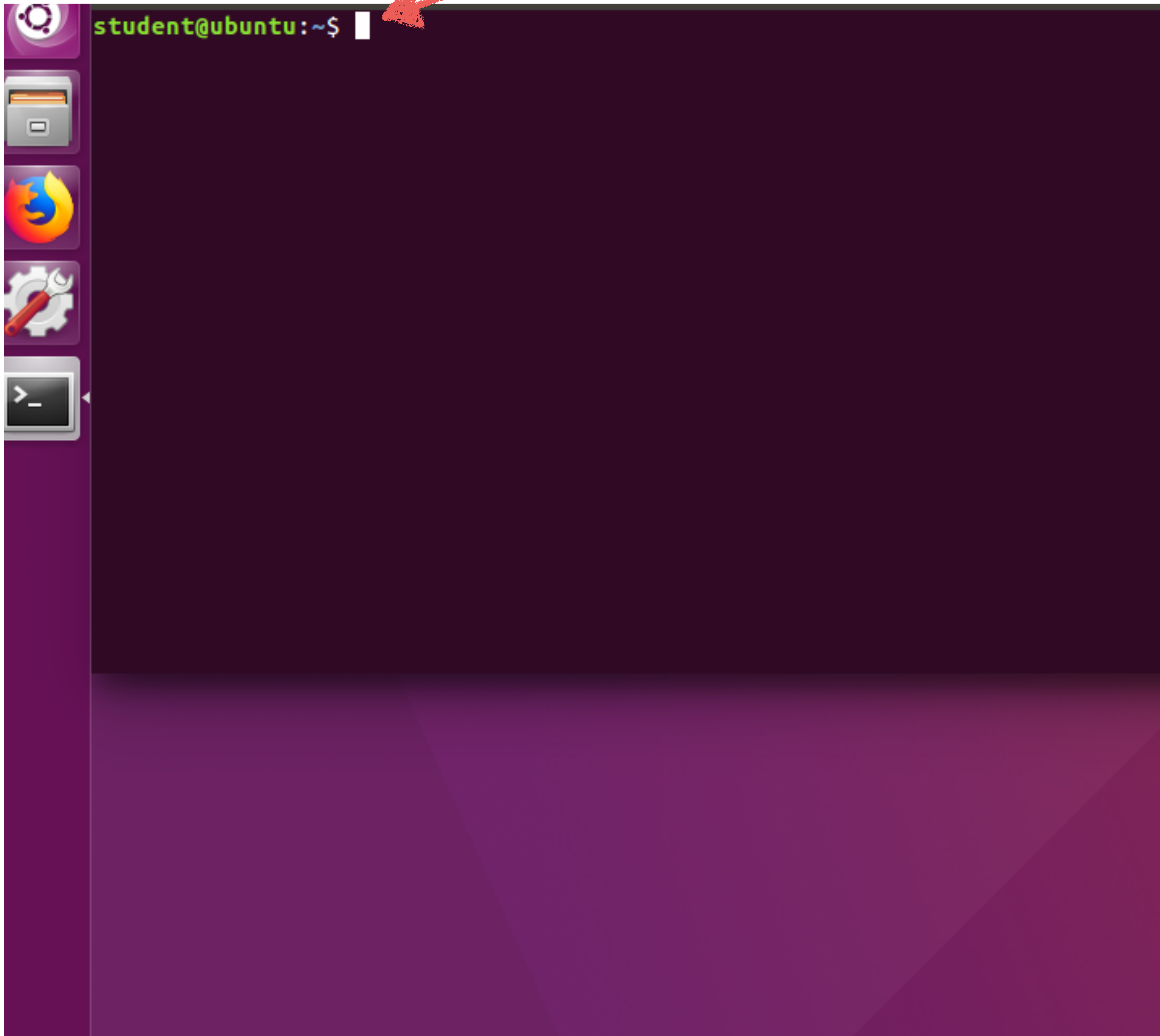
In order to configure certain management and networking features, VMware Workstation needs to know if this virtual machine was moved or copied.

If you don't know, answer "I Copied It".

Click here



Open a terminal



```
student@ubuntu:~$ 
```

First step with the terminal

- The *shell*: a command line interpreter to interact with the kernel. It can be started using the "*terminal*" application. There are many of them the default is called "*bash*" for "Bourne-Again shell".

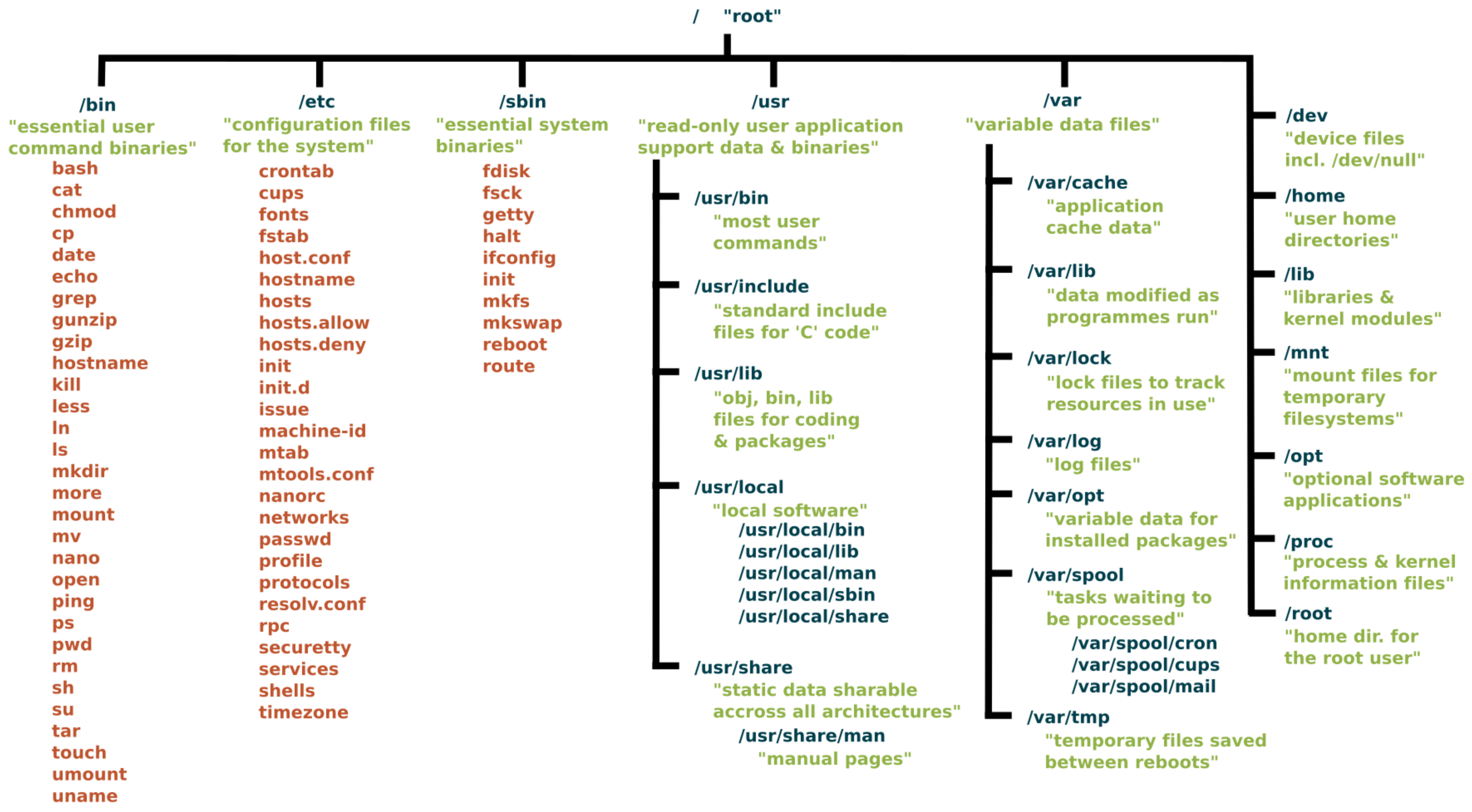
- Some commands:

  - ★ **pwd**: Print working directory will tell you what directory you are in.
  - ★ **cd** *path*– Change Directory
  - ★ **ls** – Lists files & directories
  - ★ **w** – display who is logged in and what they are doing
  - ★ **rm** *filename* – Remove the file named filename
  - ★ **mkdir** *dirname* - make directories

Filesystem

- Everything in Linux is either a file or a directory
- The Filesystem Hierarchy Standard (FSH) is the way that these files and directories are structured:
  - ★ / is the root directory
  - ★ /bin: Essential user command executables to perform operations: copy, show directory, …
  - ★ /boot
  - ★ /dev: devices
  - ★ /etc: Configuration files for all programs
  - ★ /lib – Essential shared libraries and kernel modules
  - ★ …

Linux filesystem hierarchy

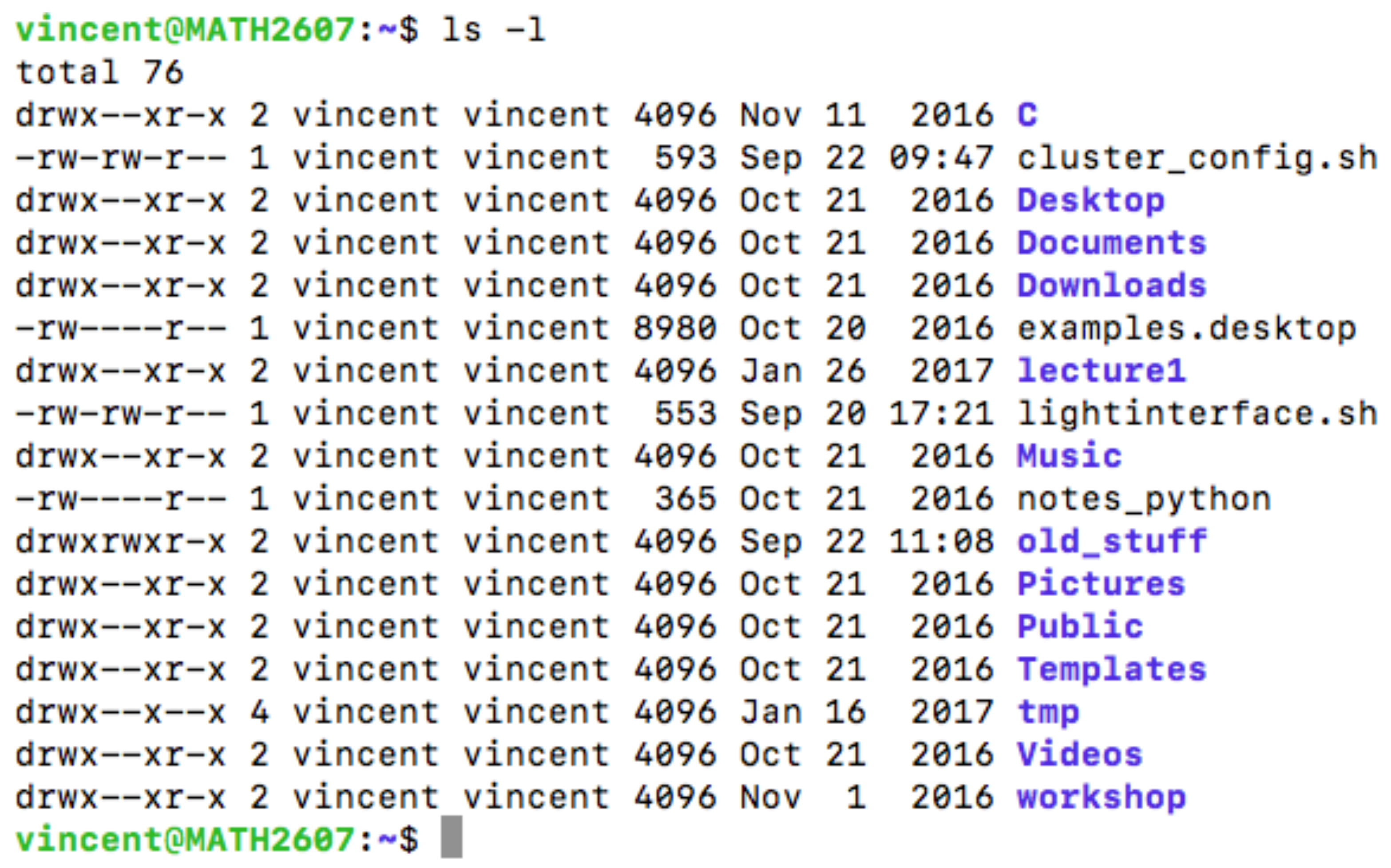

/ "root"
/bin "essential user command binaries"
bash cat chmod cp date echo grep gunzip gzip hostname kill less ln ls mkdir more mount mv nano open ping ps pwd rm sh su tar touch umount uname
/etc "configuration files for the system"
crontab cups fonts fstab host.conf hostname hosts hosts.allow hosts.deny init init.d issue machine-id mtab mtools.conf nanorc networks passwd profile protocols resolv.conf rpc securetty services shells timezone
/sbin "essential system binaries"
fdisk fsck getty halt ifconfig init mkfs mkswap reboot route
/usr "read-only user application support data & binaries"
/usr/bin "most user commands"
/usr/include "standard include files for 'C' code"
/usr/lib "obj, bin, lib files for coding & packages"
/usr/local "local software"
/usr/local/bin /usr/local/lib /usr/local/man /usr/local/sbin /usr/local/share
/usr/share "static data sharable accross all architectures"
/usr/share/man "manual pages"
/var "variable data files"
/var/cache "application cache data"
/var/lib "data modified as programmes run"
/var/lock "lock files to track resources in use"
/var/log "log files"
/var/opt "variable data for installed packages"
/var/spool "tasks waiting to be processed"
/var/spool/cron /var/spool/cups /var/spool/mail
/var/tmp "temporary files saved between reboots"
/dev "device files incl. /dev/null"
/home "user home directories"
/lib "libraries & kernel modules"
/mnt "mount files for temporary filesystems"
/opt "optional software applications"
/proc "process & kernel information files"
/root "home dir. for the root user"

Source: https://linuxfoundation.org/blog/classic-sysadmin-the-linux-filesystem-explained/attachment/standard-unix-filesystem-hierarchy/

Example

- Example type “**ls -l**”: (-l is an option which stands for long listing format)

- You can see files in black, directory in blue, the date of last modification among other things.

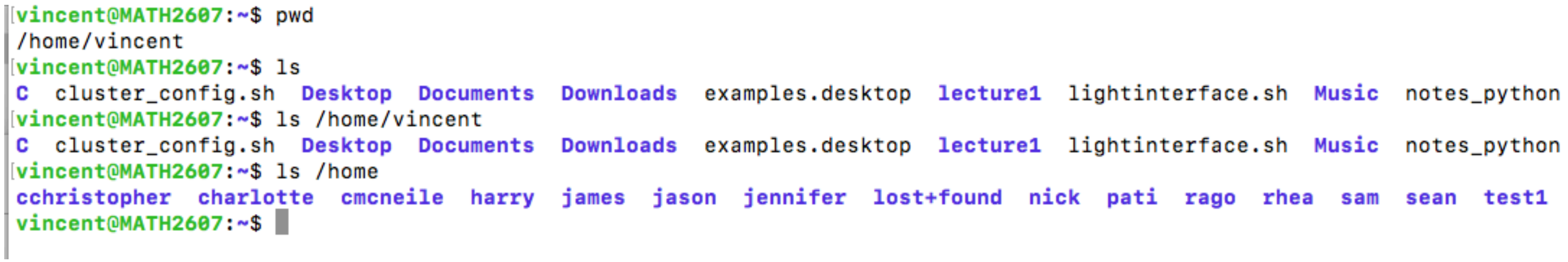


```
vincent@MATH2607:~$ ls -l
total 76
drwx--xr-x 2 vincent vincent 4096 Nov 11  2016 C
-rw-rw-r-- 1 vincent vincent  593 Sep 22 09:47 cluster_config.sh
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Desktop
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Documents
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Downloads
-rw----r-- 1 vincent vincent 8980 Oct 20  2016 examples.desktop
drwx--xr-x 2 vincent vincent 4096 Jan 26  2017 lecture1
-rw-rw-r-- 1 vincent vincent  553 Sep 20 17:21 lightinterface.sh
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Music
-rw----r-- 1 vincent vincent  365 Oct 21  2016 notes_python
drwxrwxr-x 2 vincent vincent 4096 Sep 22 11:08 old_stuff
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Pictures
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Public
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Templates
drwx--x--x 4 vincent vincent 4096 Jan 16  2017 tmp
drwx--xr-x 2 vincent vincent 4096 Oct 21  2016 Videos
drwx--xr-x 2 vincent vincent 4096 Nov  1  2016 workshop
vincent@MATH2607:~$ 
```

Example

- Current location: pwd

- Examples:



```
vincent@MATH2607:~$ pwd
/home/vincent
vincent@MATH2607:~$ ls
C  cluster_config.sh  Desktop  Documents  Downloads  examples.desktop  lecture1  lightinterface.sh  Music  notes_python
vincent@MATH2607:~$ ls /home/vincent
C  cluster_config.sh  Desktop  Documents  Downloads  examples.desktop  lecture1  lightinterface.sh  Music  notes_python
vincent@MATH2607:~$ ls /home
cchristopher  charlotte  cmcneile  harry  james  jason  jennifer  lost+found  nick  pati  rago  rhea  sam  sean  test1
vincent@MATH2607:~$ 
```

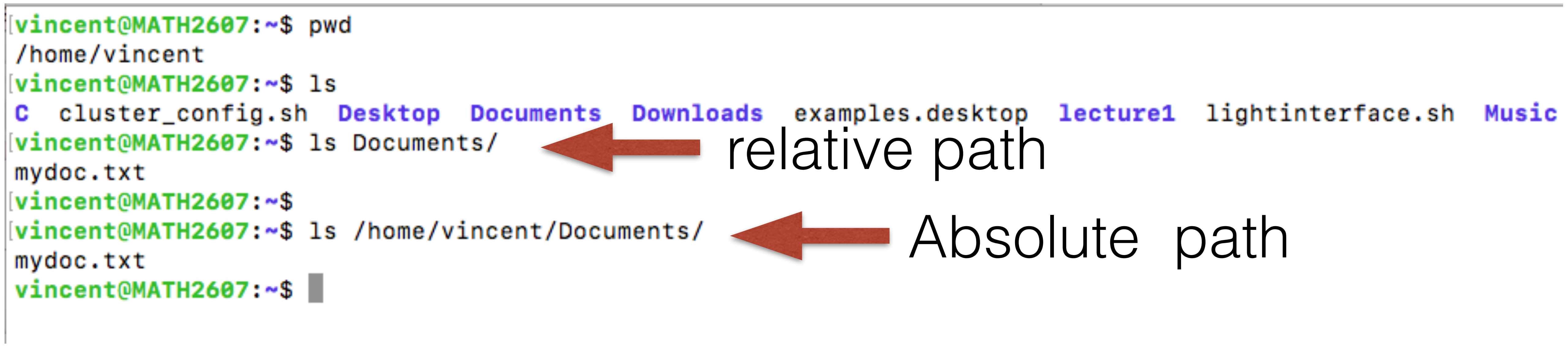
- Note that ls return what is inside the directory where you are

- ls /home return what is in /home:

- A *path* is a means to get to a particular file or directory on the system.

More on paths

- Path can be **absolute** or **relative:**

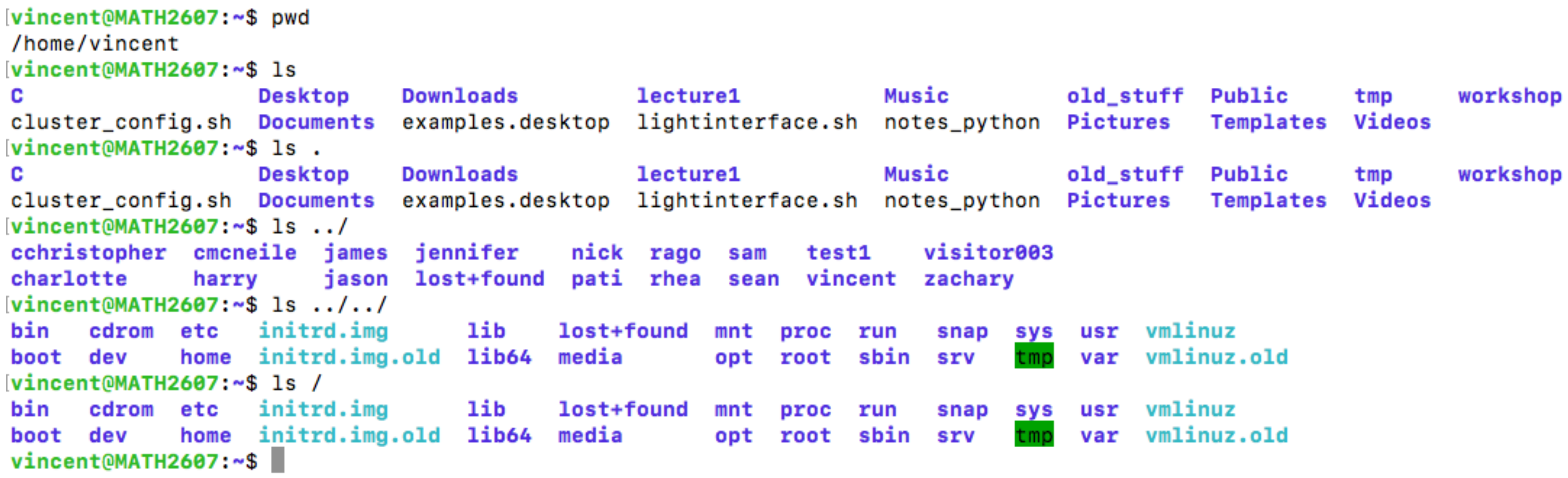


```
vincent@MATH2607:~$ pwd
/home/vincent
vincent@MATH2607:~$ ls
C  cluster_config.sh  Desktop  Documents  Downloads  examples.desktop  lecture1  lightinterface.sh  Music
vincent@MATH2607:~$ ls Documents/
mydoc.txt
vincent@MATH2607:~$
vincent@MATH2607:~$ ls /home/vincent/Documents/
mydoc.txt
vincent@MATH2607:~$
```

- Shortcuts:
  - ★ **~** (tilde) - shortcut for your home directory: ls ~
  - ★ **.** (dot) - This is a reference to your current directory. Try: *ls ./Documents*
  - ★ **..** (dotdot)- This is a reference to the parent directory. Try: *ls ..*

Example

- Playing a bit with ls:



```
vincent@MATH2607:~$ pwd
/home/vincent
vincent@MATH2607:~$ ls
C                  Desktop    Downloads         lecture1            Music          old_stuff  Public     tmp     workshop
cluster_config.sh  Documents  examples.desktop  lightinterface.sh   notes_python   Pictures   Templates  Videos
vincent@MATH2607:~$ ls .
C                  Desktop    Downloads         lecture1            Music          old_stuff  Public     tmp     workshop
cluster_config.sh  Documents  examples.desktop  lightinterface.sh   notes_python   Pictures   Templates  Videos
vincent@MATH2607:~$ ls ../
cchristopher  cmcneile  james  jennifer    nick  rago  sam   test1    visitor003
charlotte     harry     jason  lost+found  pati  rhea  sean  vincent  zachary
vincent@MATH2607:~$ ls ../../
bin   cdrom  etc   initrd.img      lib    lost+found  mnt  proc  run   snap  sys  usr  vmlinuz
boot  dev    home  initrd.img.old  lib64  media       opt  root  sbin  srv   tmp  var  vmlinuz.old
vincent@MATH2607:~$ ls /
bin   cdrom  etc   initrd.img      lib    lost+found  mnt  proc  run   snap  sys  usr  vmlinuz
boot  dev    home  initrd.img.old  lib64  media       opt  root  sbin  srv   tmp  var  vmlinuz.old
vincent@MATH2607:~$ 
```

A few remarks

- Linux is an “extensionless” file system (you can name an executable file.txt if you want… But it would be mean)

- Luckily there is a command to determining what type of file a particular file is:

  ★ **file**-- obtain information about what type of file a file or directory is.

- The manual:

  ★ **man <command>** - Look up the manual page for a particular command. (e.g try man ls) [press q to quit the manual, space or up and down arrows to scroll down]

  ★ Type **/-l** in the ls manual to look for “-l”

  ★ then **n** to jump to the next iteration

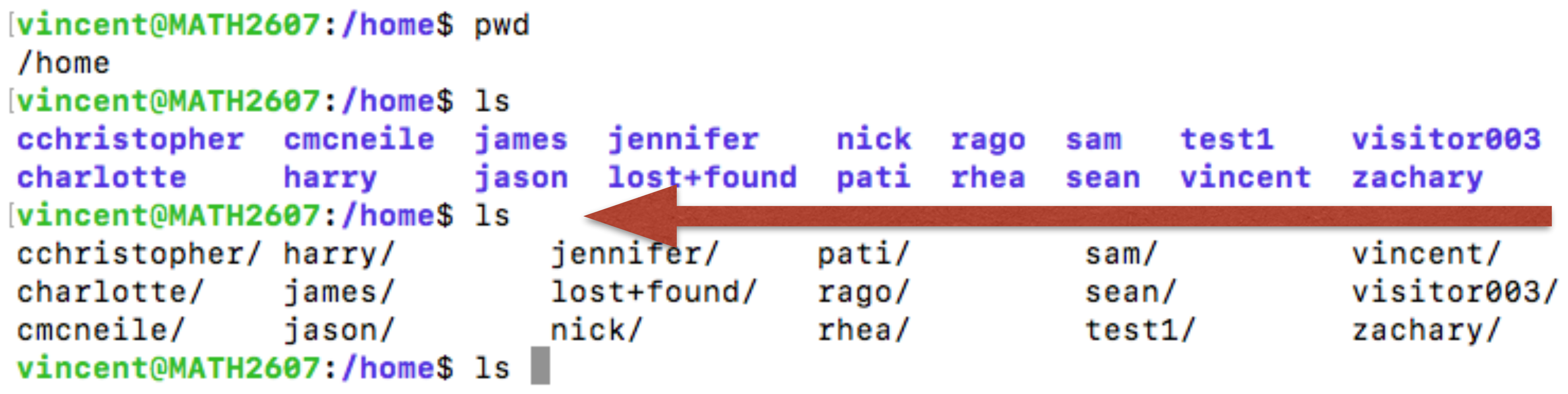
  ★ Look at the ls option : -a,-l,-r,-t for instance

Creating directories & files

- To create a directory: try in your home folder: *mkdir testdir*
- To remove it : *rmdir testdir* (note that testdir must be empty for rmdir to work)
- To create an empty file called “testfile” you can type *touch testfile*
- To copy “testfile" in Documents type *cp testfile ./Documents*
- To copy “testfile” to another place and give it the name “new” type *:*
  cp testfile new
- To move “testfile” to a directory type *mv testfile ./Documents*
- To move “new” to a new file called “new2” type *mv new new2*
- You can then remove it by typing *rm new2*

Tab completion

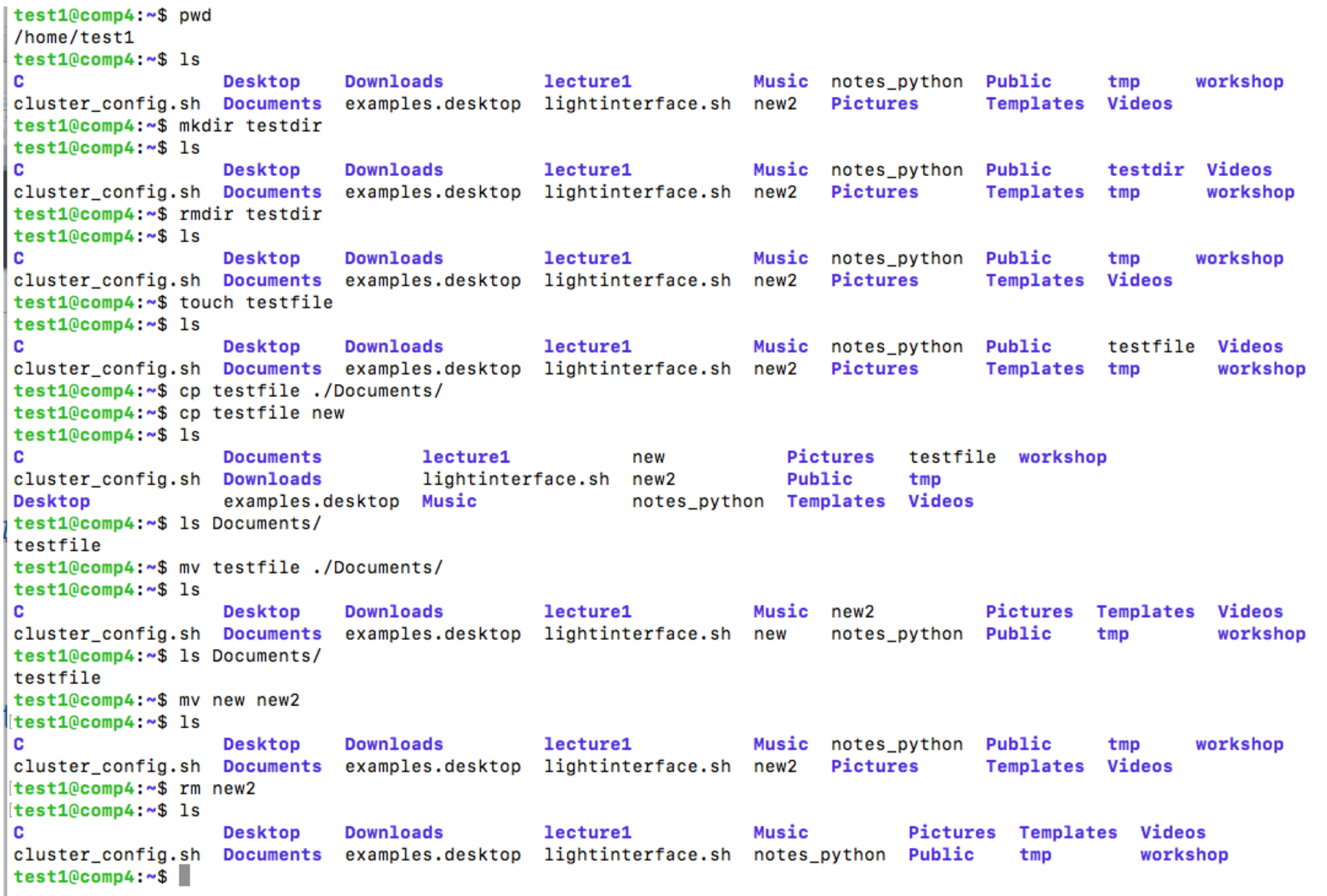
- Typing path can be tedious…
- When you start typing a path (anywhere on the command line, you're not just limited to certain commands) you may hit the Tab key on your keyboard at any time which will invoke an auto complete action. If nothing happens then that means there are several possibilities. If you hit Tab again it will show you those possibilities.
- E.g



```
vincent@MATH2607:/home$ pwd
/home
vincent@MATH2607:/home$ ls
cchristopher  cmcneile  james  jennifer    nick  rago  sam   test1    visitor003
charlotte     harry     jason  lost+found  pati  rhea  sean  vincent  zachary
vincent@MATH2607:/home$ ls
cchristopher/ harry/      jennifer/    pati/       sam/        vincent/
charlotte/    james/      lost+found/  rago/       sean/       visitor003/
cmcneile/     jason/      nick/        rhea/       test1/      zachary/
vincent@MATH2607:/home$ ls
```

I typed twice tab

Creating directories & files



```
test1@comp4:~$ pwd
/home/test1
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music  notes_python  Public     tmp     workshop
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  new2   Pictures      Templates  Videos
test1@comp4:~$ mkdir testdir
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music  notes_python  Public     testdir  Videos
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  new2   Pictures      Templates  tmp      workshop
test1@comp4:~$ rmdir testdir
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music  notes_python  Public     tmp     workshop
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  new2   Pictures      Templates  Videos
test1@comp4:~$ touch testfile
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music  notes_python  Public     testfile  Videos
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  new2   Pictures      Templates  tmp       workshop
test1@comp4:~$ cp testfile ./Documents/
test1@comp4:~$ cp testfile new
test1@comp4:~$ ls
C                  Documents         lecture1           new           Pictures   testfile  workshop
cluster_config.sh  Downloads         lightinterface.sh  new2          Public     tmp
Desktop            examples.desktop  Music              notes_python  Templates  Videos
test1@comp4:~$ ls Documents/
testfile
test1@comp4:~$ mv testfile ./Documents/
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music  new2          Pictures  Templates  Videos
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  new    notes_python  Public    tmp        workshop
test1@comp4:~$ ls Documents/
testfile
test1@comp4:~$ mv new new2
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music  notes_python  Public     tmp     workshop
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  new2   Pictures      Templates  Videos
test1@comp4:~$ rm new2
test1@comp4:~$ ls
C                  Desktop    Downloads         lecture1           Music         Pictures  Templates  Videos
cluster_config.sh  Documents  examples.desktop  lightinterface.sh  notes_python  Public    tmp        workshop
test1@comp4:~$ 
```

Displaying the contents of a file

- Most basic commands :
  - ★ **cat -** reads files sequentially, writing them to standard output
  - ★ **more** - to view (but not modify) the contents of a text file on screen [q to quit, space to scroll down, b to scroll back]
- Try to display the content of :
  - ★ The file hostname in **/etc**
  - ★ The file cp in /bin
  - ★ The file syslog in /var/log
- Editors :
  - ★ nedit, gedit : simple to use [linux]
  - ★ nano: text editor using a command line interface [linux]
  - ★ Emacs or xemacs: text editor (*) [Linux, Windows, macOS]
  - ★ Vi,Vim(*) [Linux, Windows, macOS]
  - ★ Atom, Visual Studio Code [Linux, Windows, macOS]

(*) very efficient but less user friendly?

Exercise

- Use the commands cd and ls to explore what directories are on your system and what's in them. Make sure you use a variety of relative and absolute paths. Some interesting places to look at are:
  - ★ /etc - Stores config files for the system.
  - ★ /var/log - Stores log files for various system programs.
  - ★ /bin - The location of several commonly used programs (e.g ls, cd are there if you look carefully !!)

File permissions

- In Linux, every file and directory belongs to a user and a group. Files and directories have a set of permissions controlling who can **read**, **write**, or **execute** them

- To see the files in the current directory along with their permissions, use the command: ls -l

- The first position indicates directory or file.
  The next positions are in groups of 3 corresponding to **user**, **group**, **other**.

- Give a file read and write permissions for group members: chmod g+rw file.txt Remove execution permissions for other (non-group) users: chmod o-x file.txt

- Give read permissions for all users (user, group, other): chmod a+r file.txt

- Give execution permissions for the user (owner) and group :chmod ug+x file.txt

- To change ownership: chown user1:group1 file.txt

- Files with execute permission are usually programs. These can be "executed" by typing the name of the file including the path. S

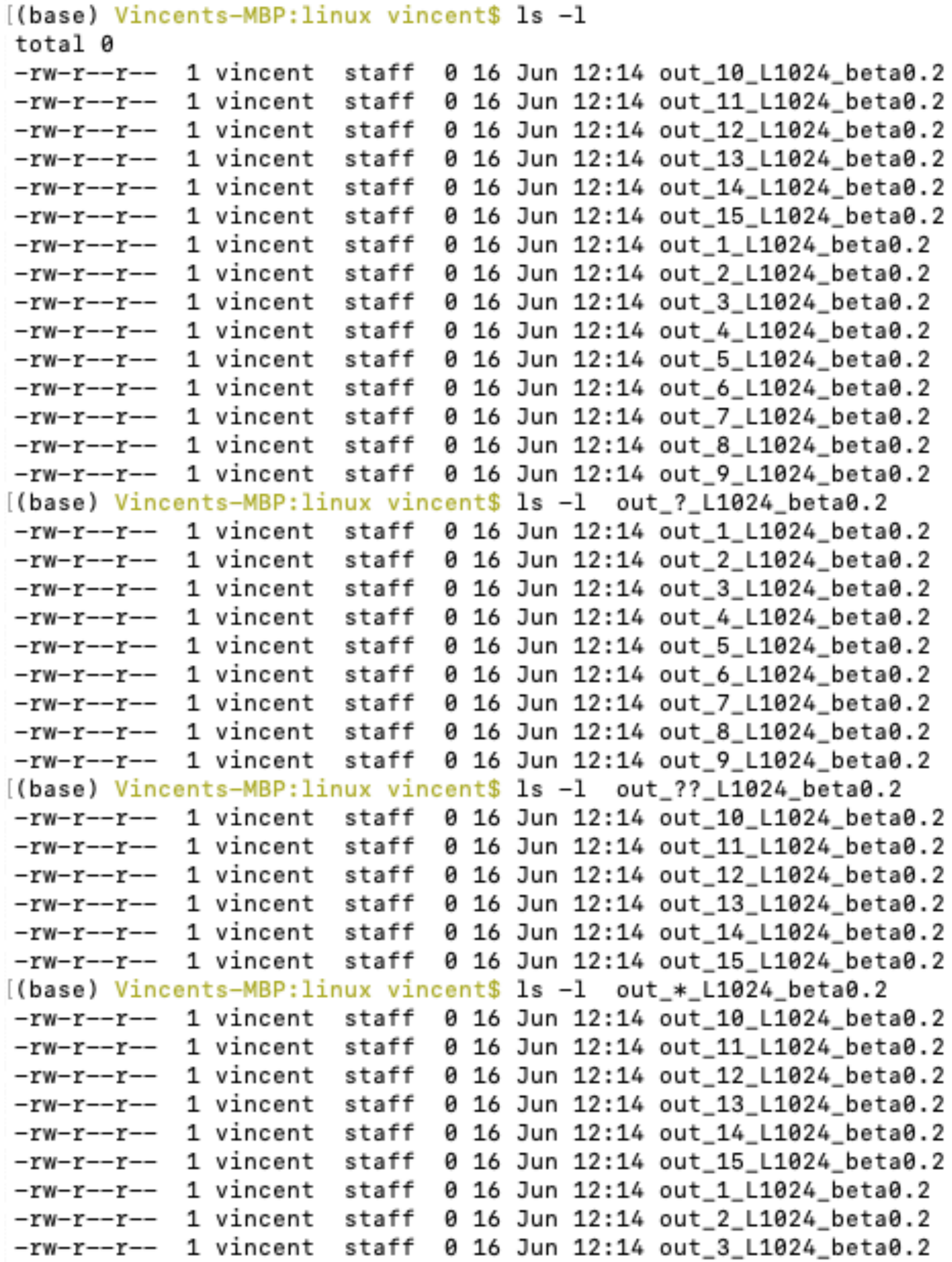
Other useful commands

- du: to see the size of a file (du -h - for “human readable format”)
- diff: to compare two files
- history: displays the list of commands previously typed by the user
- tar, gzip: to compress files
- wc: word, line and character count
- which: locate a program file in the user's path
- grep: file pattern searcher
- tail/head: to display the head/last part of a file
- top, ps , kill: to list the running processes
- date: return the date/time on the system
- sed, awk: stream editor/text processing.

More e.g on: https://linux.die.net

Wildcards

- There are lots of “tricks” to be efficient.
- A **wildcard** in Linux is a symbol or a set of symbols that stands in for other characters.
- Examples :
    - ★ ? Matches a single character
    - ★ * matches any character or set of characters, including no character



```
[(base) Vincents-MBP:linux vincent$ ls -l
 total 0
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_10_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_11_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_12_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_13_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_14_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_15_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_1_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_2_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_3_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_4_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_5_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_6_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_7_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_8_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_9_L1024_beta0.2
[(base) Vincents-MBP:linux vincent$ ls -l  out_?_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_1_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_2_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_3_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_4_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_5_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_6_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_7_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_8_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_9_L1024_beta0.2
[(base) Vincents-MBP:linux vincent$ ls -l  out_??_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_10_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_11_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_12_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_13_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_14_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_15_L1024_beta0.2
[(base) Vincents-MBP:linux vincent$ ls -l  out_*_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_10_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_11_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_12_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_13_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_14_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_15_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_1_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_2_L1024_beta0.2
 -rw-r--r--  1 vincent  staff  0 16 Jun 12:14 out_3_L1024_beta0.2
```

Standard output/Standard error

- Usually command return their output in the terminal is called the standard output.

- If there is an error in the command execution it typically return through another stream called the standard error.

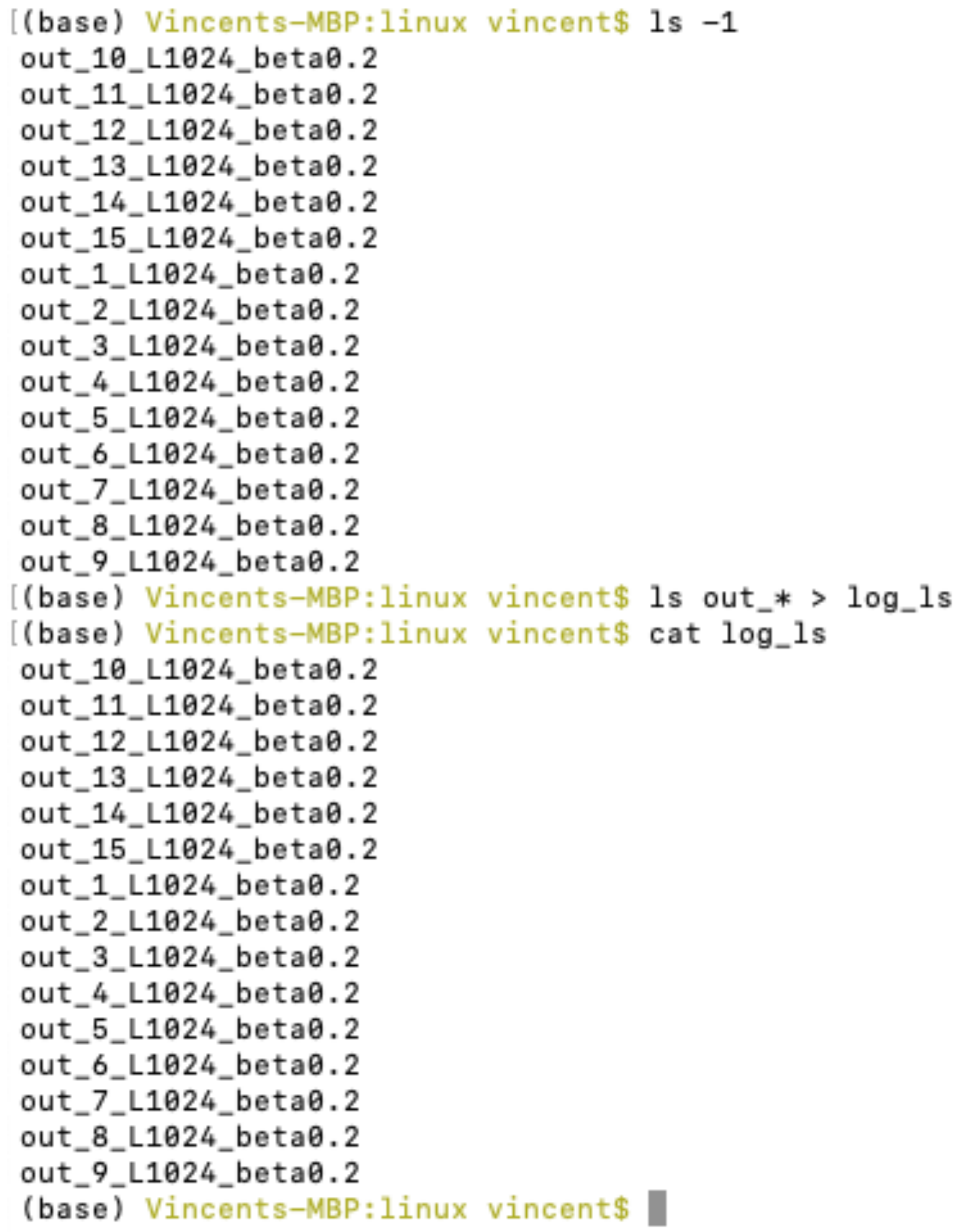
- From a practical point of view, it is sometimes useful to redirect the standard output and/or the standard error into a file that is stored and can be looked at later.

- In short :

    ★ > redirects output to a file, overwriting the file.

    ★ >>redirects output to a file appending the redirected output at the end.

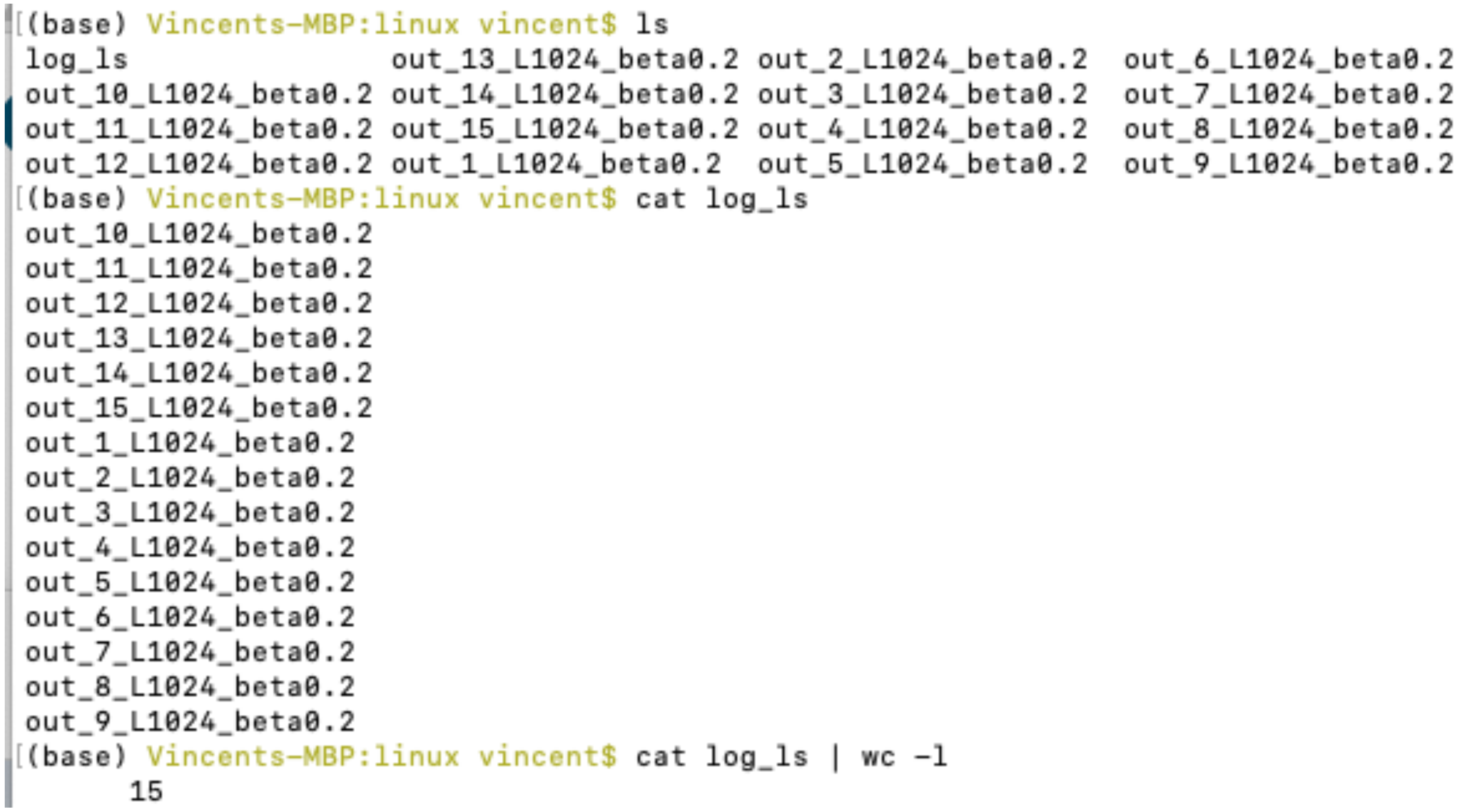
    ★ >& (or 2>&1) for redirecting standard output and standard to a file



```
[(base) Vincents-MBP:linux vincent$ ls -1
out_10_L1024_beta0.2
out_11_L1024_beta0.2
out_12_L1024_beta0.2
out_13_L1024_beta0.2
out_14_L1024_beta0.2
out_15_L1024_beta0.2
out_1_L1024_beta0.2
out_2_L1024_beta0.2
out_3_L1024_beta0.2
out_4_L1024_beta0.2
out_5_L1024_beta0.2
out_6_L1024_beta0.2
out_7_L1024_beta0.2
out_8_L1024_beta0.2
out_9_L1024_beta0.2
[(base) Vincents-MBP:linux vincent$ ls out_* > log_ls
[(base) Vincents-MBP:linux vincent$ cat log_ls
out_10_L1024_beta0.2
out_11_L1024_beta0.2
out_12_L1024_beta0.2
out_13_L1024_beta0.2
out_14_L1024_beta0.2
out_15_L1024_beta0.2
out_1_L1024_beta0.2
out_2_L1024_beta0.2
out_3_L1024_beta0.2
out_4_L1024_beta0.2
out_5_L1024_beta0.2
out_6_L1024_beta0.2
out_7_L1024_beta0.2
out_8_L1024_beta0.2
out_9_L1024_beta0.2
(base) Vincents-MBP:linux vincent$ ▌
```

Combining commands

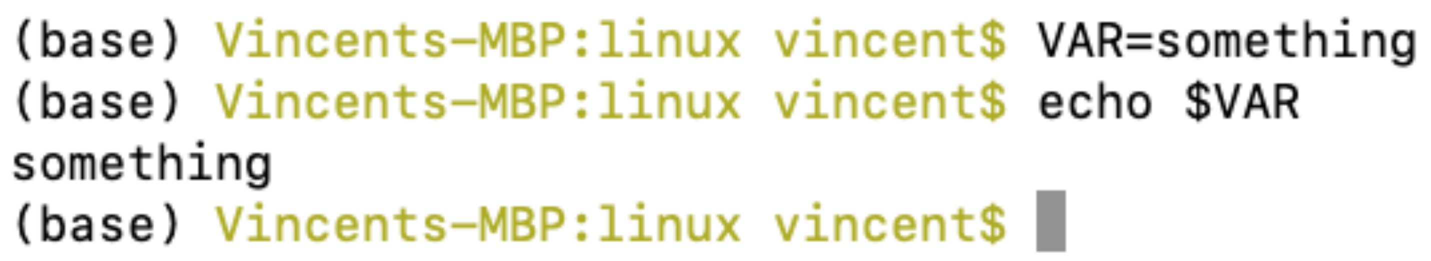
- The pipe command | : allows to send the output of one command to another



```
[(base) Vincents-MBP:linux vincent$ ls
log_ls                  out_13_L1024_beta0.2 out_2_L1024_beta0.2  out_6_L1024_beta0.2
out_10_L1024_beta0.2 out_14_L1024_beta0.2 out_3_L1024_beta0.2  out_7_L1024_beta0.2
out_11_L1024_beta0.2 out_15_L1024_beta0.2 out_4_L1024_beta0.2  out_8_L1024_beta0.2
out_12_L1024_beta0.2 out_1_L1024_beta0.2  out_5_L1024_beta0.2  out_9_L1024_beta0.2
[(base) Vincents-MBP:linux vincent$ cat log_ls
out_10_L1024_beta0.2
out_11_L1024_beta0.2
out_12_L1024_beta0.2
out_13_L1024_beta0.2
out_14_L1024_beta0.2
out_15_L1024_beta0.2
out_1_L1024_beta0.2
out_2_L1024_beta0.2
out_3_L1024_beta0.2
out_4_L1024_beta0.2
out_5_L1024_beta0.2
out_6_L1024_beta0.2
out_7_L1024_beta0.2
out_8_L1024_beta0.2
out_9_L1024_beta0.2
[(base) Vincents-MBP:linux vincent$ cat log_ls | wc -l
      15
```

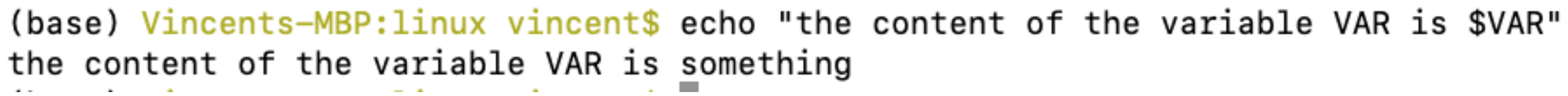
Variables in bash

- A bash variable can contain a number, a character, a string of characters.
- No need to declare the variable. The assignment creates the variable.
- To display a variable use the echo command:, eg:



```
(base) Vincents-MBP:linux vincent$ VAR=something
(base) Vincents-MBP:linux vincent$ echo $VAR
something
(base) Vincents-MBP:linux vincent$ 
```

- Or



```
(base) Vincents-MBP:linux vincent$ echo "the content of the variable VAR is $VAR"
the content of the variable VAR is something
```

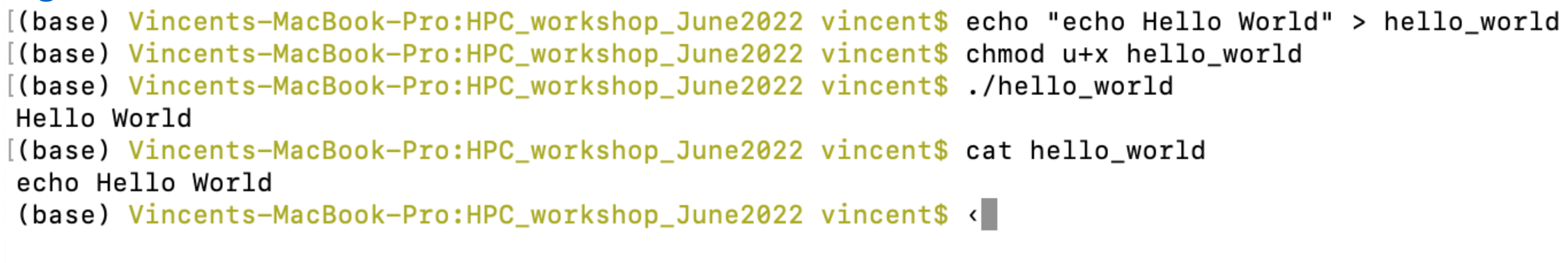
- In most cases : `$var` and `${var}` are the same (The braces are only needed to resolve ambiguity in expressions)
- To define a variable that contains the output of a command : `var=$(date)`

Environment variables

- Environment variables are a set of variable that allows to customise the behaviour of the OS.
- For instance when you type `'ls'`, the system look for the command ls in the $PATH variable.
- Type: `'echo $HOME'` or `'echo $PATH'`.
- Most programs and libraries (e.g python, R, MPI implementations) have specific environment variables that control their behaviour. This is often a source of issues (e.g software Y cannot find library Z while I just spent 2 days to install it…)

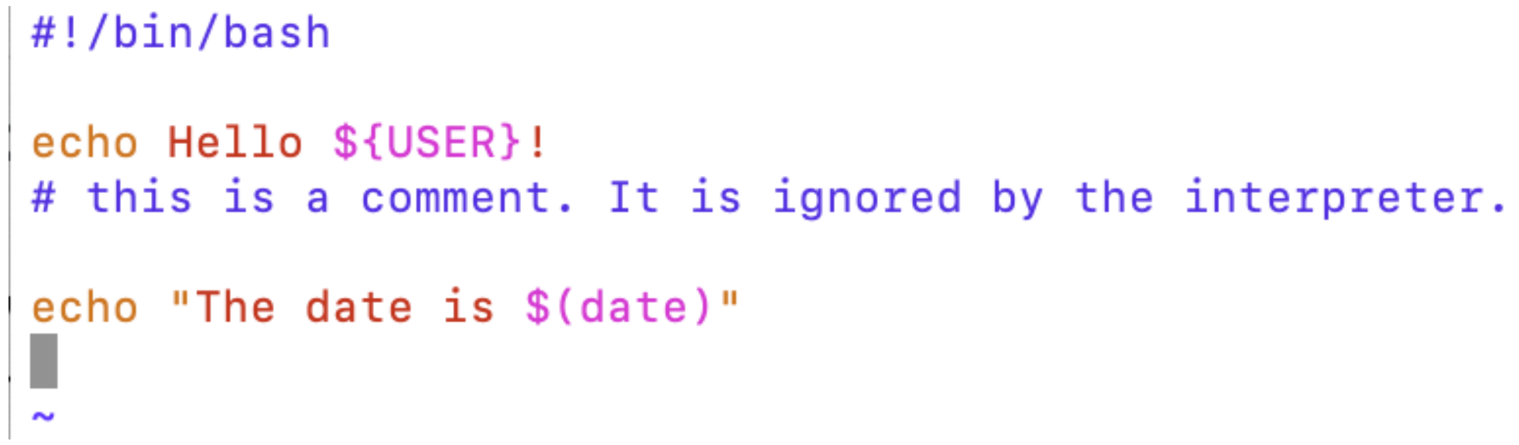
Scripts

- Eg:



```
[(base) Vincents-MacBook-Pro:HPC_workshop_June2022 vincent$ echo "echo Hello World" > hello_world
[(base) Vincents-MacBook-Pro:HPC_workshop_June2022 vincent$ chmod u+x hello_world
[(base) Vincents-MacBook-Pro:HPC_workshop_June2022 vincent$ ./hello_world
Hello World
[(base) Vincents-MacBook-Pro:HPC_workshop_June2022 vincent$ cat hello_world
echo Hello World
(base) Vincents-MacBook-Pro:HPC_workshop_June2022 vincent$ 
```

- To instruct a shell to run your script in a certain shell, add #!/bin/bash on the first line of the script.

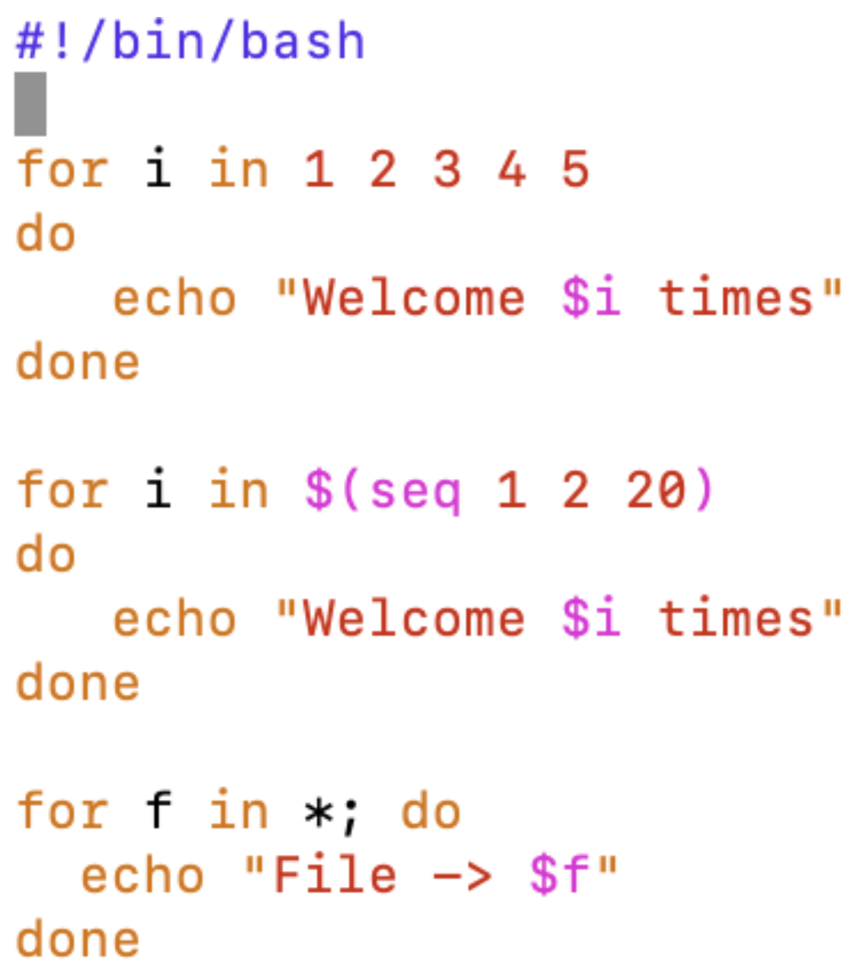


```
#!/bin/bash

echo Hello ${USER}!
# this is a comment. It is ignored by the interpreter.

echo "The date is $(date)"

~
```

- All line starting with # are comments : they are ignored by the shell interpreter

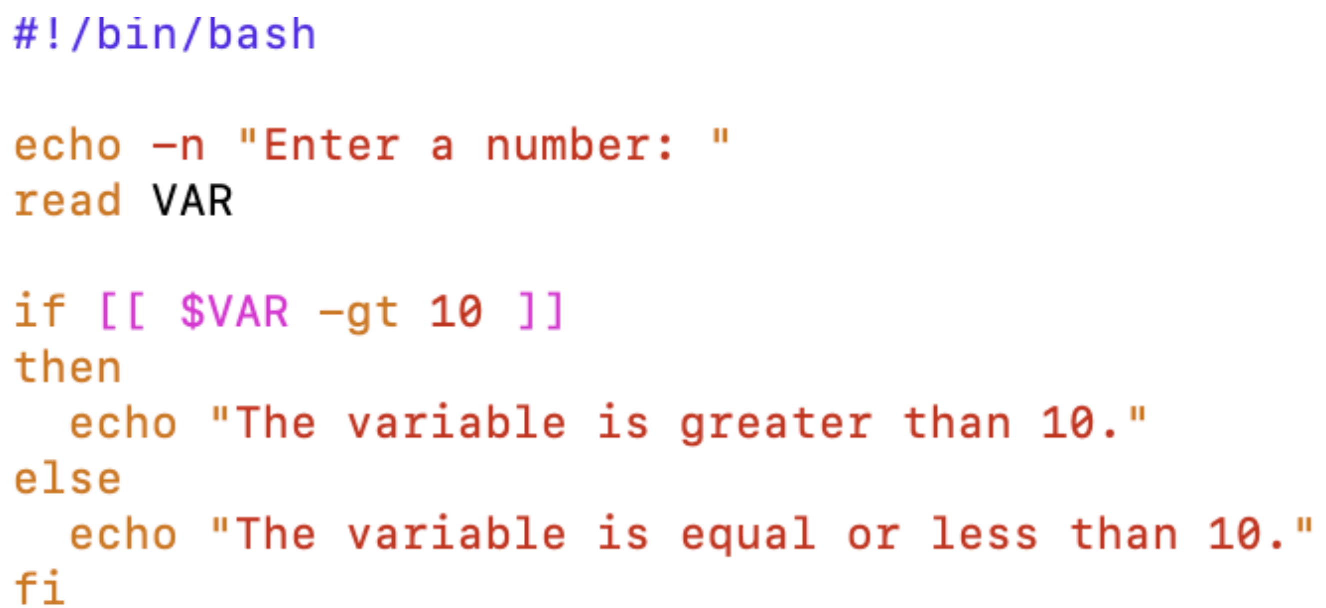
Slightly more sophisticated examples

- For loops:



```
#!/bin/bash

for i in 1 2 3 4 5
do
   echo "Welcome $i times"
done

for i in $(seq 1 2 20)
do
   echo "Welcome $i times"
done

for f in *; do
  echo "File -> $f"
done
```

- If … Else



```
#!/bin/bash

echo -n "Enter a number: "
read VAR

if [[ $VAR -gt 10 ]]
then
  echo "The variable is greater than 10."
else
  echo "The variable is equal or less than 10."
fi
```

Exercises

- Find commands that give you the following information
    - ★ How many cores are available
    - ★ How much space is left on the hard disk
    - ★ How much RAM has the computer
    - ★ Which version of ubuntu are you running
    - ★ How much space take your home directory
- Write a script that creates a file called backup_yourusername_hour:minutes_day_month_year.tar.gz which contains all files in your home directory
- Where are the password of each user stored ?

HPC environment

foseres (late 2016)

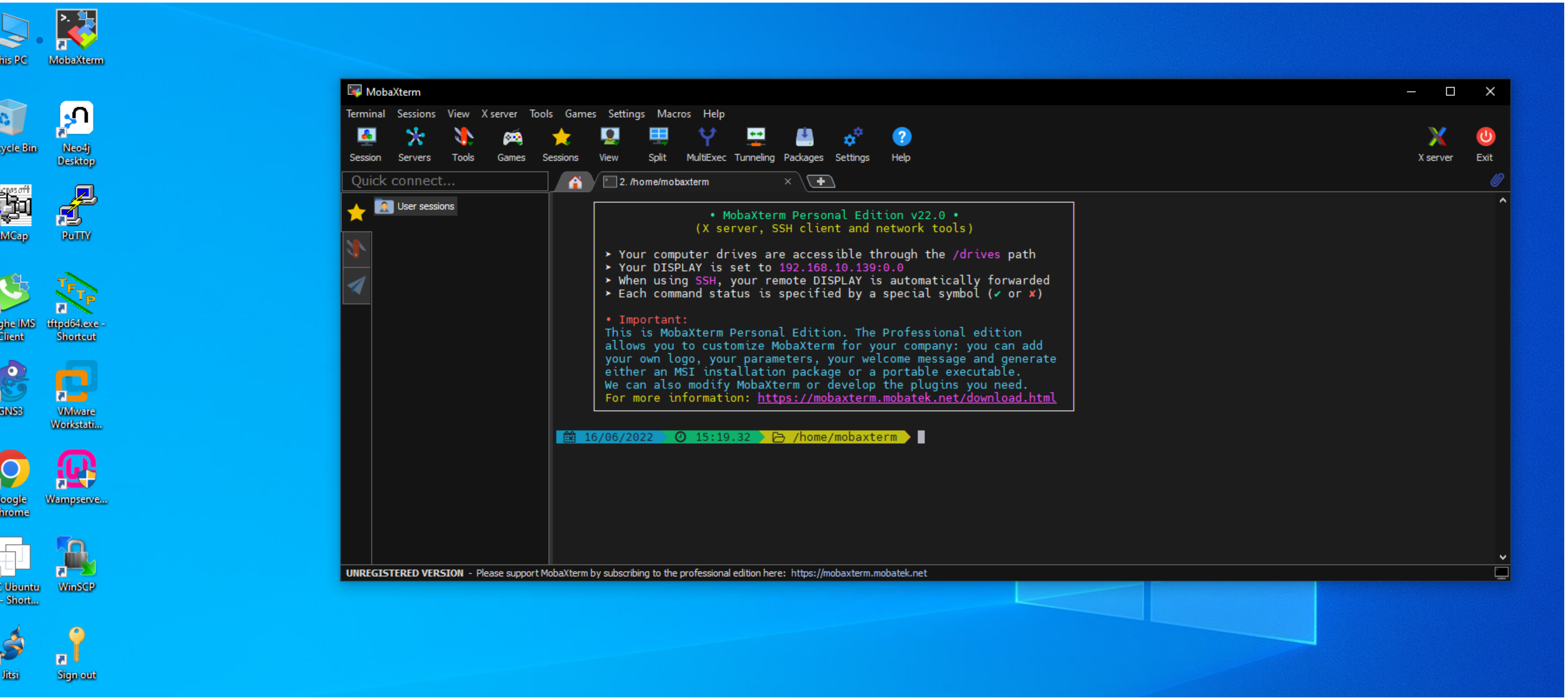
- Computing node: two 2.1 GHz, 16-core Intel Xeon E5-2683v4 (Broadwell) series processors.
- 48 nodes= 1536 cores
- Memory: 128 GB per node
- Network: Intel OPA fabric [100 GB/s]
- Storage: 0.5 PB - Lustre parallel file system designed to give high read/write bandwidth for parallel I/O operations.

Remote access

Remote access from Windows

- **Start MobaXterm**

- **The folder ./MyDocuments corresponds to the folder Documents in Windows.**



MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
2. /home/mobaxterm
User sessions

```
                 • MobaXterm Personal Edition v22.0 •
               (X server, SSH client and network tools)

 ➤ Your computer drives are accessible through the /drives path
 ➤ Your DISPLAY is set to 192.168.10.139:0.0
 ➤ When using SSH, your remote DISPLAY is automatically forwarded
 ➤ Each command status is specified by a special symbol (✔ or ✘)

 • Important:
 This is MobaXterm Personal Edition. The Professional edition
 allows you to customize MobaXterm for your company: you can add
 your own logo, your parameters, your welcome message and generate
 either an MSI installation package or a portable executable.
 We can also modify MobaXterm or develop the plugins you need.
 For more information: https://mobaxterm.mobatek.net/download.html

 16/06/2022  15:19.32  /home/mobaxterm
```

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Connection to foseres

- **Secure Shell** (**SSH**): cryptographic network protocol for operating network services securely over an unsecured network.

- Syntax:

```
ssh  username@foseres.fost.plymouth.ac.uk
```

- You will be asked to change your password:
  `#length=10 #Digits=4 #Lower case=2 #Upper case=1 #Special character=1`

- To come back to your computer just **logout** of the main server.

Copy files over ssh

- Command to copy from/to a remote host: **scp** (secure copy)
- Syntax:

```
scp Source Destination
```

- Examples:

```
scp SourceFile user@foseres.fost.plymouth.ac.uk:./

scp SourceFile user@foseres.fost.plymouth.ac.uk:./directory/TargetFile .
```

To copy a directory to your home directory:

```
scp -r dir user@foseres.fost.plymouth.ac.uk:./
```

To copy from the remote host to your local machine in the present directory:

```
scp user@foseres.fost.plymouth.ac.uk:./directory/File .
```

- There are other tools to copy files over ssh. A useful one is rsync.
- Exercise: Copy a file from your Windows machine to foseres

An HPC machine is flexible tool

- **We provide and maintain a flexible computing environment.**
- **Each user must fine tune the system to its need**
- The system admin team provides a number of pre-configured compilers/software/libraries, that can be loaded using a specific command.

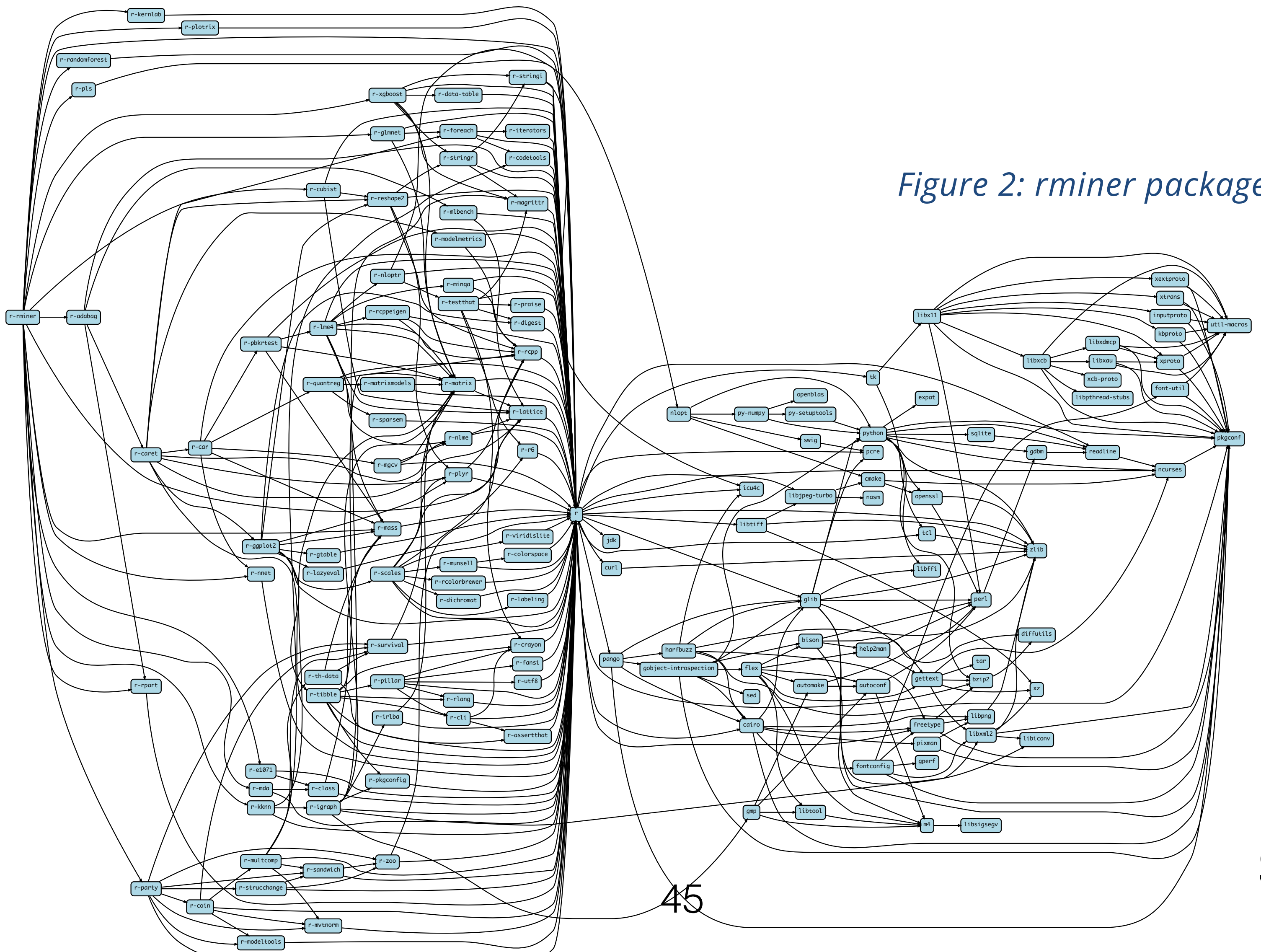


Figure 2: rminer package, with 150 dependencies.

Source: llnl (spack)

Modules

The module command

- To list all currently available module type:

```
module avail
```

- If you partially know the name of the module your are looking for type

```
module spider gsl
```

In that case you should see : gsl/2.4 gsl/2.6 gsl/2.7

If you type “module spider gsl/2.7” you find that it requires “ intel/18.0.1”, “intel/18.0.1.163” or intel/19.0.5.281

If you type module spider gsl/2.6 you find that it requires “ gnu8/8.3.0”

- To load a module type

```
module load ModuleName
```

The command will configure the shell for an application by modifying the environment variables

- To find help on a given module

```
module help ModuleName
```

The module command

- If you want to start from a fresh environment type:

```
module purge
```

- Many user will need a compiler, I currently advise one of the following

```
module load  gnu8/8.3.0   #-> this is the gnu8 compiler
module load compiler/2021.2.0 #-> this is the intel_oneAPI compiler.
```

- If you need MPI support, use:

```
module load openmpi3/3.1.4 # -> if you have loaded the gnu8 compiler
Module load mpi/2021.2.0 # ->if you have load the intel_oneAPI
compiler
```

- Exercise load your favourite module.

Job scheduler

The job scheduler

- An HPC machine have many users. There is always a specific software in charge of scheduling job on the “computing nodes”. In our case the software is called Slurm (an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters)

  More on https://slurm.schedmd.com/quickstart.html
- Slurm is used on many of the world’s top 500 supercomputers.

- As a cluster workload manager, Slurm has three key functions.
  1. It allocates exclusive and/or non-exclusive access to resources (compute nodes) to users for some duration of time so they can perform work.
  2. It provides a framework for starting, executing, and monitoring work (normally a parallel job) on the set of allocated nodes.
  3. It arbitrates contention for resources by managing a queue of pending work.

- Good material at https://hpc.llnl.gov/training/tutorials#training materials. See also https://en.wikipedia.org/wiki/Slurm_Workload_Manager

- The most important command are:

```
sbatch : to submit a script (more on that later)
squeue: to look at the queue (who is running, who is waiting to run…)
sinfo: to find out how many job are available
```

- Exercise: find how many node are idling, allocated, and down (now)

- There are two types of queues : The test queue (2 nodes) and the normal queue (the remaining 46 nodes)

SLURM

Slurm consists of a slurmd daemon running on each compute node and a central slurmctld daemon running on a management node (with optional fail-over twin).
Important user commands include:

command	description
sacct	report job accounting information about active or completed jobs
salloc	allocate resources for a job in real time
sbatch	submit a job script for later execution (the script typically contains one or more srun commands to launch parallel tasks)
scancel	cancel a pending or running job
sinfo	reports the state of partitions and nodes managed by Slurm (it has a variety of filtering, sorting, and formatting options)
squeue	reports the state of jobs (it has a variety of filtering, sorting, and formatting options), by default, reports the running jobs in priority order followed by the pending jobs in priority order
srun	used to submit a job for execution in real time

SLURM

- Jobs typically specify what resources are needed, such as type of machine, number of machines, job duration, amount of memory required, account to charge, etc.

- The jobs are submitted to the execution queue with the commands sbatch <script file>.
The queueing system prints a number (the job id) almost immediately and returns control to the linux prompt. At this point the job is in the submission queue.

- Once the job is submitted, it will sit in a pending state until the resources have been allocated to the job. The progress of the job can be monitored using the squeue command.

Job script to reserve 1 entire node

```
#!/bin/bash

#SBATCH -J test               # Job name
#SBATCH --partition normal    # Job queue - It could also be set to test
#SBATCH -o job.%j.out         # Name of stdout output file (%j expands to jobId)
#SBATCH -N 1                  # Number of nodes
#SBATCH -n 1                  # Number of MPI tasks
#SBATCH -t 00:03:00           # Run time (hh:mm:ss) - 3 minutes

# setting up (example)
module load gnu8
module load  R/3.6.1

# preprocessing
# …
# print out some info for the logs

echo “Current directory $(pwd)”
echo “Starting: $(date)”

ls -l

# actual computational intensive step
# Launch serial executable:

Rscript hello.R # or R CMD BATCH hello.R

# post processing / cleaning up

echo “Job complete: $(date)”
```

Note: this script is available in /tmp/Rjob_example

Note: the script hello.R is available in /tmp/hello.R
Copy it to the folder which contains the batch script.

Exercise: Adapt the script to run your own hello world program in the language of your choice.

SLURM

squeue is the main command for monitoring the state of systems, groups of jobs or individual jobs. scontrol also provides some features for monitoring jobs. The command squeue prints the list of current jobs. The list looks something like this:

```
vdrach@MATH2607:~$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                62    normal     test   vdrach PD       0:00      5 (Resources)
                63    normal     test   vdrach PD       0:00      6 (Priority)
                64    normal     test   vdrach PD       0:00      7 (Priority)
                65    normal     test   vdrach PD       0:00      8 (Priority)
                60    normal     test   vdrach  R       0:03      3 comp[4-6]
                61    normal     test   vdrach  R       0:01      4 comp[1-3,7]
```

List all current jobs for a user	squeue -u <username>
List all running jobs for a user	squeue -u <username> -t RUNNING
List all pending jobs for a user	squeue -u <username> -t PENDING
List all current jobs in the general partition for a user	squeue -u <username> -p general
List detailed information for a job (useful for troubleshooting)	scontrol show jobid -dd <jobid>

SLURM

scancel and scontrol are the main commands for controlling the jobs.

To cancel one job	scancel <jobid>
To cancel all the jobs for a user	scancel -u <username>
To cancel all the pending jobs for a user	scancel -t PENDING -u <username>
To cancel one or more jobs by name	scancel –name myJobName
To pause a particular job	scontrol hold <jobid>
To resume a particular job	scontrol resume <jobid>
To requeue (cancel and rerun) a particular job	scontrol requeue <jobid>

Comments/Good practices

- The more resources your request, the longer you will usually wait.
- The wall time is limited to 3 days. If you need more let us know.
- Use the test queue (#-p test) to test your scripts if they can be run on 1 or 2 nodes.
- You can find information about job by typing:

```
scontrol show job JobID
```

- You can also request an interactive job to actually login on the computing node for some time and do whatever you want:

```
srun -N1 -n1 -p normal -t 00:03:00 --pty bash -i
```

Virtual environments

Virtual environments: generalities

- You can also install libraries and softwares in your $HOME directory
- A virtual environment is a convenient way to package your own software suite.
- In which case is it useful:
  1. Your favourite R/python library is not installed
  2. You have one program that has been developed for a specific python/ R library and another program written for a different version of the same library.
- There are different to create virtual environments. Here we focus on the open-source conda package manager (originally developed by Anaconda Inc, it is now a separate package released under the BSD license).

First example

- Your application require tensorflow 2.6.0 a free and open-source for machine learning and artificial intelligence, developed by the Google Brain team together with python 3.9.

```
module load conda

conda create -n py39_tf260_mkl

conda activate py39_tf260_mkl

conda search tensorflow

# conda install tensorflow=2.6.0
# Or if you want to specify a particular “build string”

conda install tensorflow=2.6.0=mkl_py39haac40d1_0


# To deactivate
conda deactivate
```

Using a virtual environment

- Now assuming your code is named my_tf2_code.py
- Add conda activate py39_tf260_mkl to your batch script before executing your python code.
- To list all the available environments:

```
conda env list
```

- To delete an environment:

```
conda env remove -n env_name
```

Comments:

- Sometimes a package is not available in the default repository of conda (the one maintained by the Anaconda Inc. team)
- conda-forge is an alternative channel. It is a community effort.
- A good place to start if you want to install a software/library is to google:
- Conda install *name*
- Sometimes it will find the package you are looking for in the Conda-forge channel , in that case:

```
conda install -c conda-forge packagename
```

- Always compare the different version available before you make your choice.
- Conda can also be used for R, Julia, …

Example with R

```
module load conda

conda create -n R

conda activate R

#conda install r-base # the default channel install R 3.6.1
#conda install -c r r-base # R 3.6.1 as well

conda install -c conda-forge r-base # R 4.1.3 s
```

- Now start R and install seamlessly the library ‘’ggplot2’’.

```
> install.packages(‘ggplot2’) # to install ggplot2 locally.
…
…
…
* DONE (ggplot2)

The downloaded source packages are in
    ‘/tmp/Rtmp5rZrsH/downloaded_packages’

Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
>
```

Exercise

- Goal: process 100 images with a program that smooth them
- Copy the folder /tmp/data and the python program /tmp/smoothing.py
- Create your virtual environment to use the program. The program use the library pillow. The syntax is the following:

```
python smoothing.py filename smoothing_parameter
```

where filename is the relative path to a jpg image and smoothing_parameter a number

- Write a slurm script that run the python program on all the images for a smoothing parameter of your choice (e.g 4.6), and make sure the smoothed images are stored in ./run_smoothing_4.6
- Ideally your script should be such that the smoothing parameter is stored in a variable and set once and for all at the beginning of the script.