Introduction to GNU/Linux

Day 1: 19/02/2024

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

MER 2030
2020-2026

With the support of the Erasmus+ Programme of the European Union

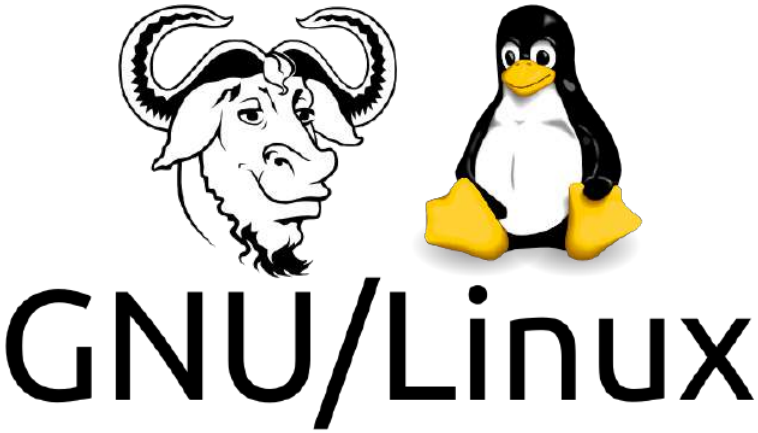
Luis J. Chueca
Postdoctoral researcher
@LuisjaChueca

Basque Centre for Climate Change (BC3)
luisjavier.chueca@bc3research.org

Sofia Marcos
Postdoctoral researcher

University of the Basque Country (UPV/EHU)
sofia.marcos@ehu.eus

Introduction



Introduction

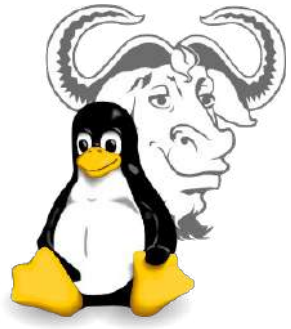
1969

UNIX®
An Open Group Standard

1983

GNU's Not Unix (GNU)

1991



GNU/LINUX

debian

1993

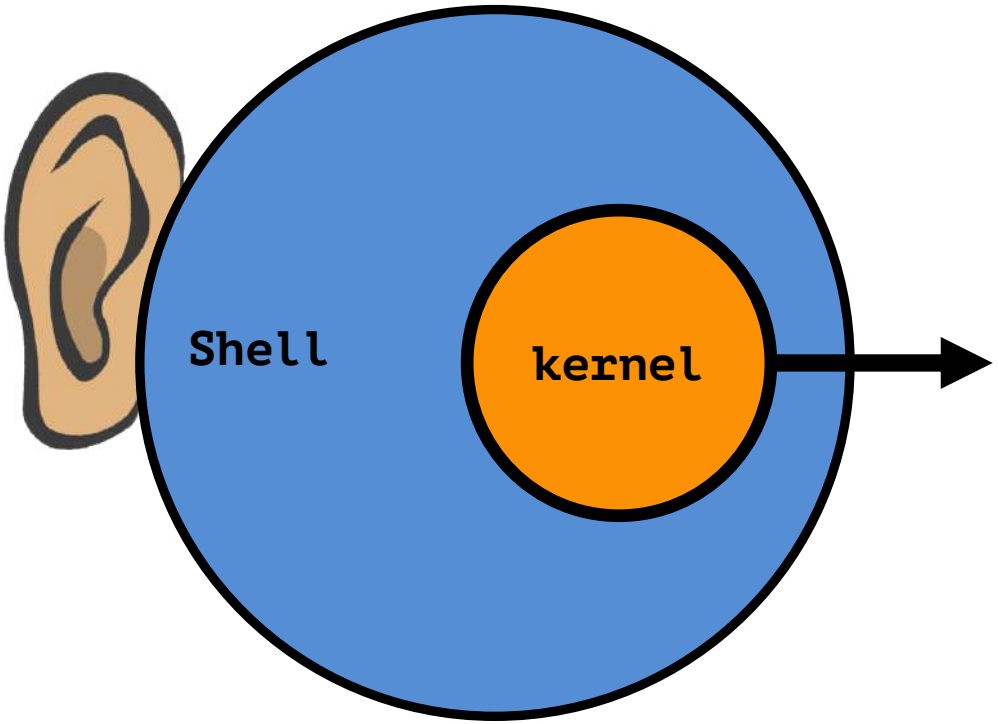
Red Hat
Enterprise Linux

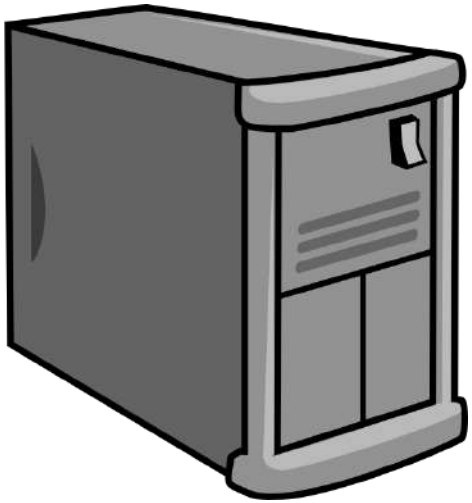
2003

Ubuntu

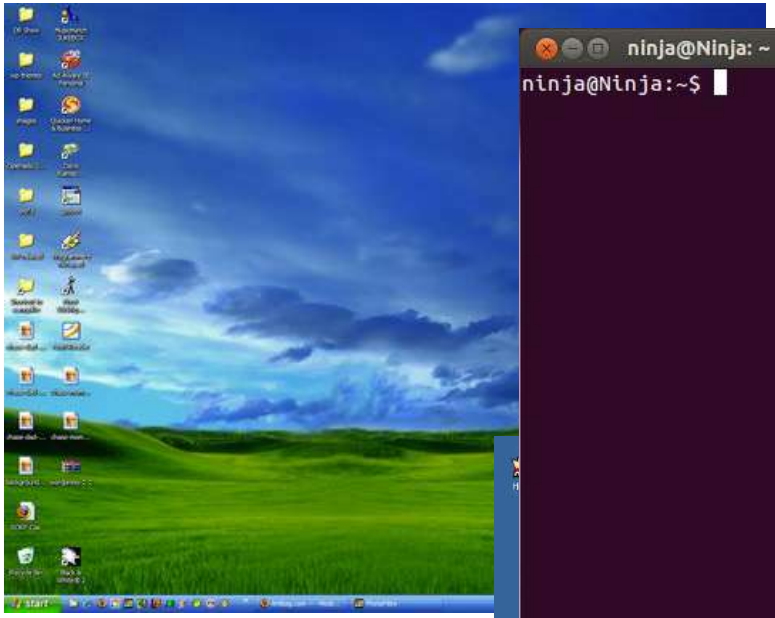
2004

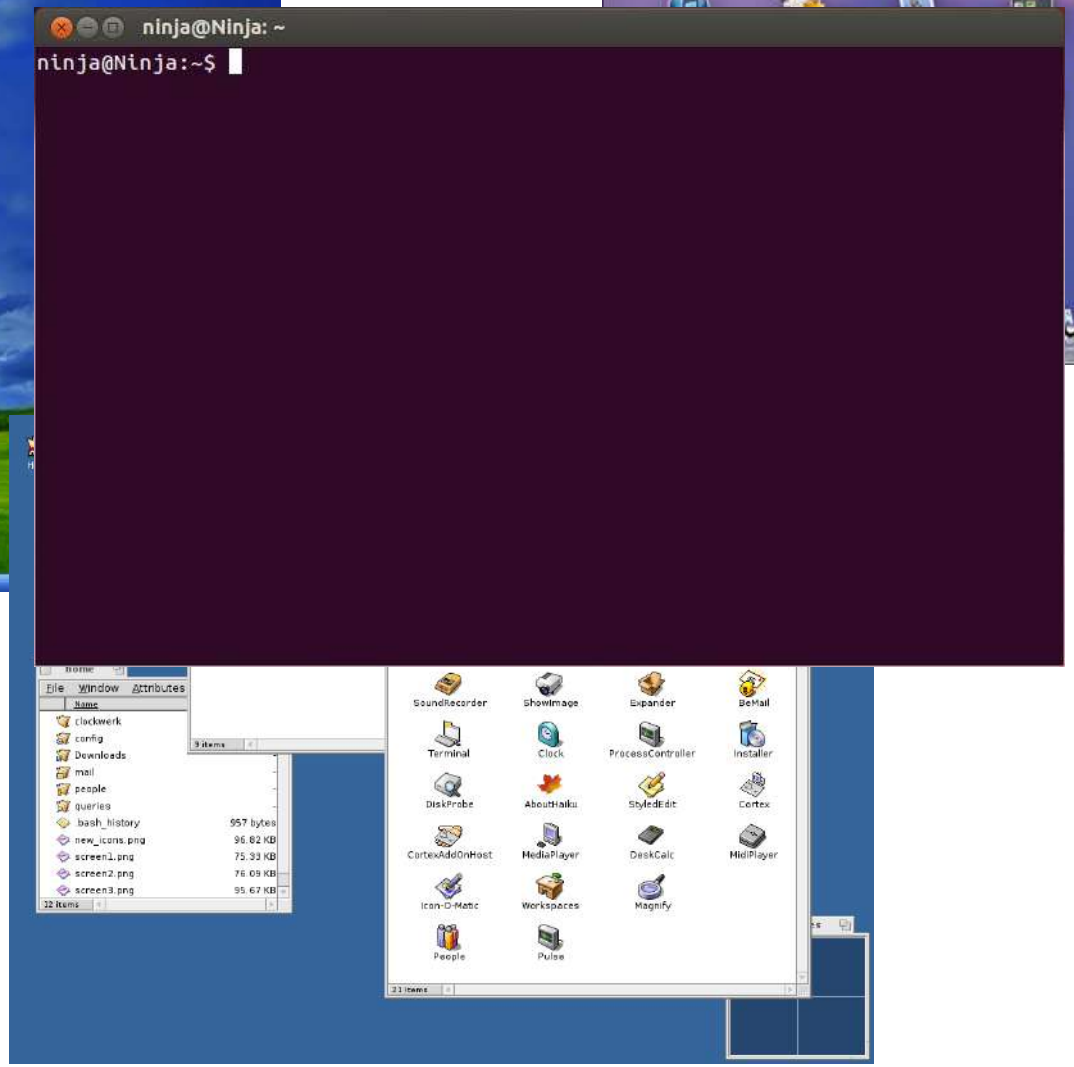
Shell concept



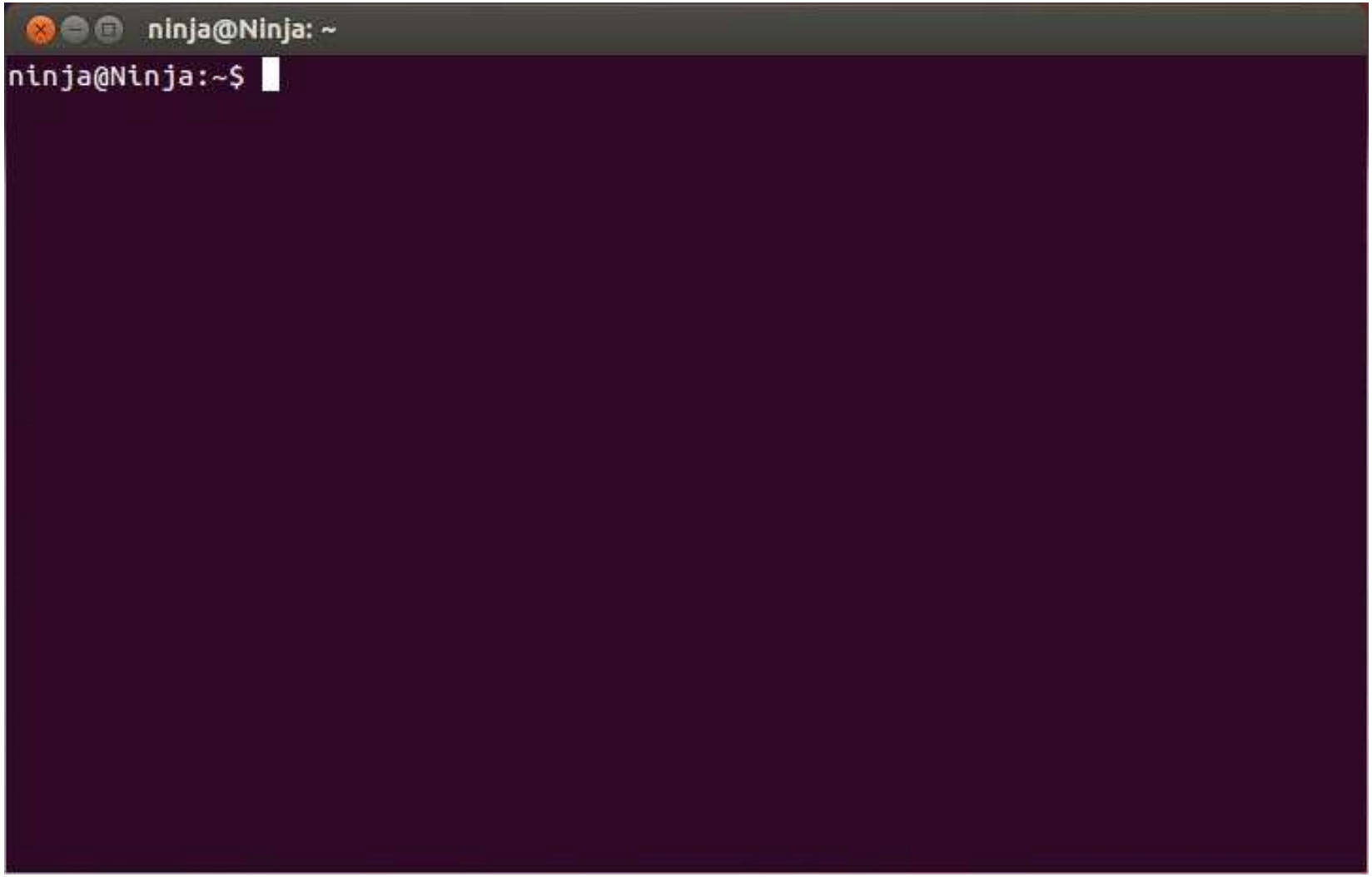


Examples of shells





The prompt

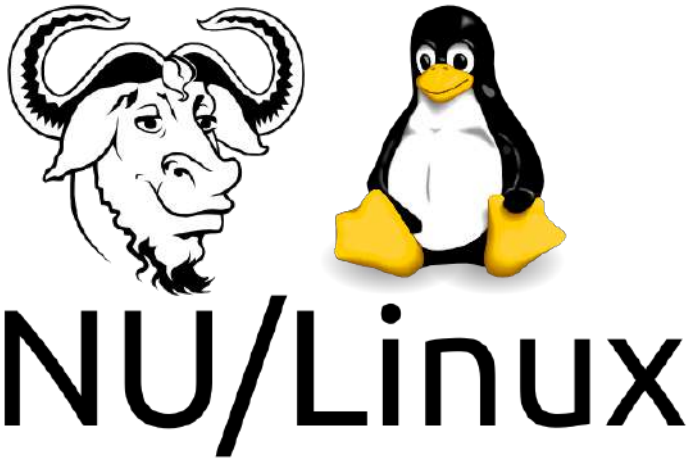


The prompt is highly configurable. Your prompt will look different than mine

Advantages of GNU/Linux

Free

Huge pool of free software



Multitasking

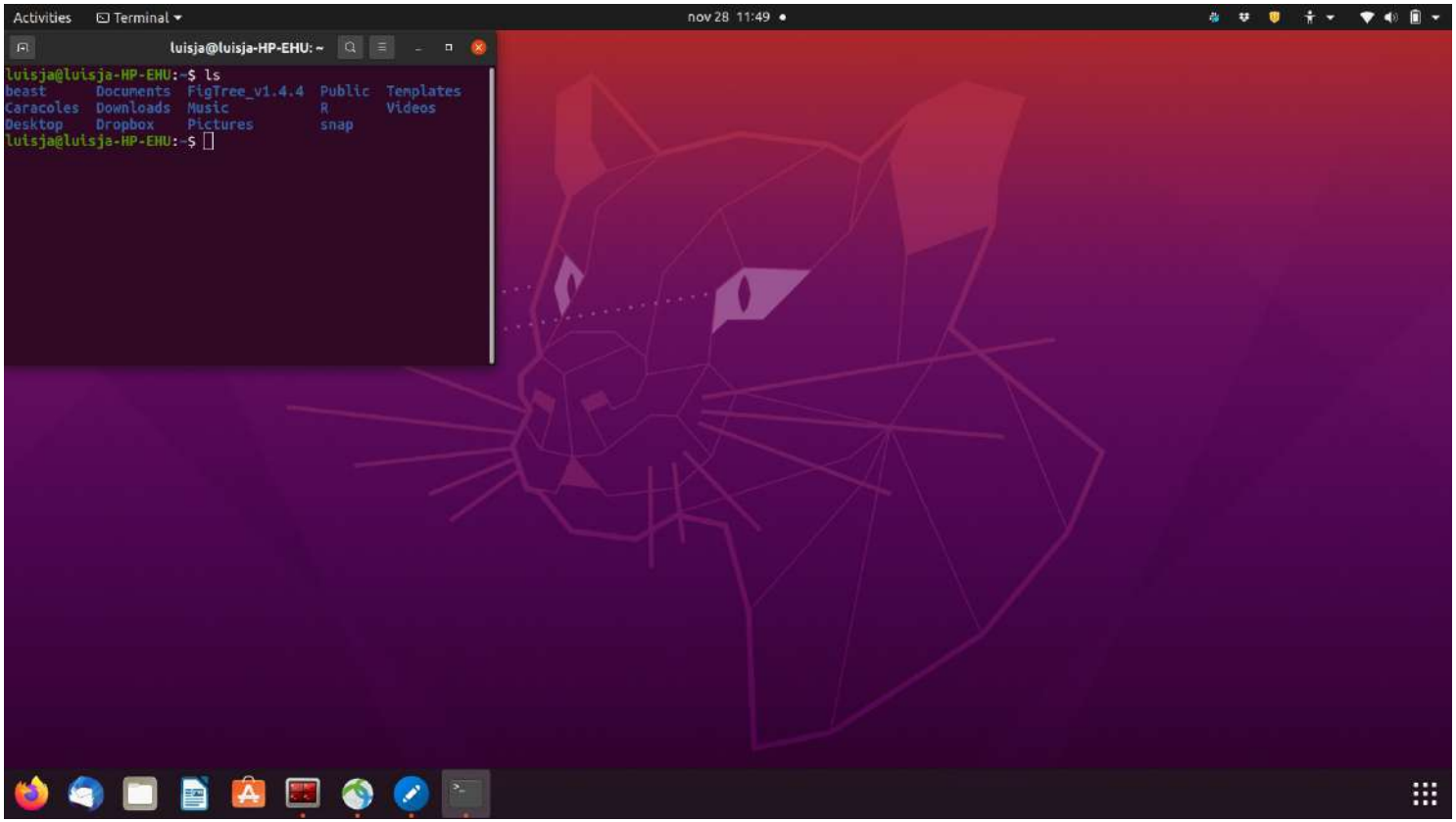
open source

High stability

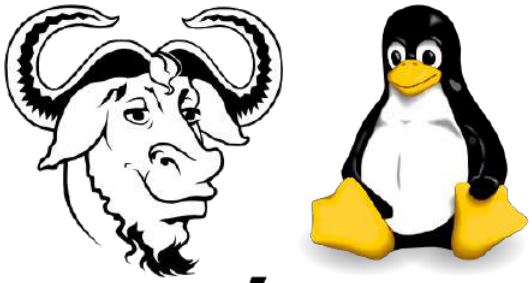
Easy administration

Multi

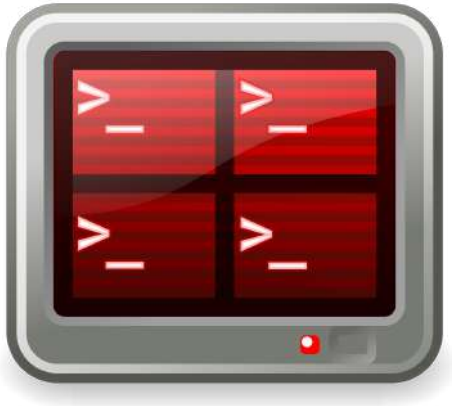
GNU/Linux



ubuntu®



GNU/Linux



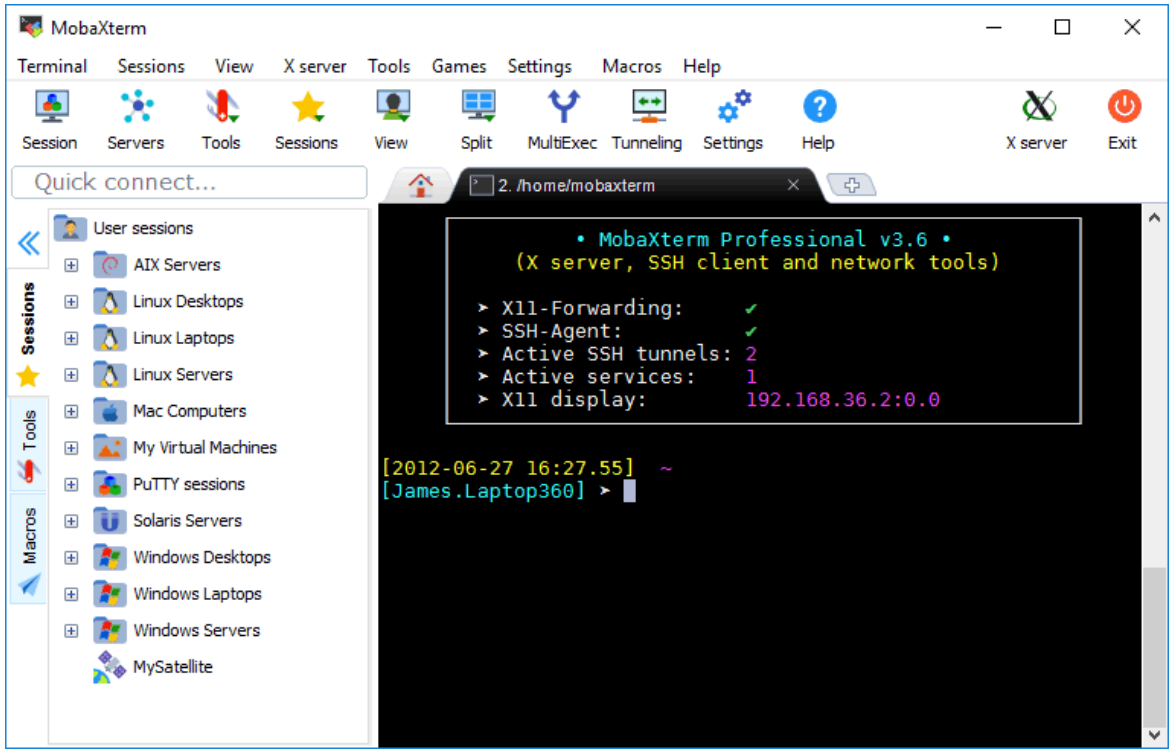
GNOME Terminator

Debian / Ubuntu
Fedora
Gentoo
(...)

MacOS X

GNU/Linux

MobaXterm



GNU/Linux

Connect to another computer

Gaming-PC:

- 8 Cores
- 16 GB RAM
- 250 GB SSD + 2 TB HDD
- Fancy graphic card
- ~ 2.000€

High-performance computer:

- 72 Cores
- 750 GB RAM
- 15 TB HDD im RAID
- ~ 15.000€

ssh connexion

```
$ ssh -Y ikasleXX@kalk2020.lgp.ehu.es
$ password:
```

GNU/Linux

ssh connexion

```
$ ssh -Y ikasleXX@kalk2020.lgp.ehu.es
$ password:
```

```
                %       %%%%%%    %%  %%%    %%        %
              %%%       %%   %%   %%  %%%%%  %%      %%%
            %% %%       %%   %%   %%  %%  %%%%%     %% %%
           %%   %%      %%%%%%    %%  %%     %%%    %%   %%
          %%%%%%%%%%    %%   %%   %%  %%      %%   %%%%%%%%%%
         %%        %%   %%    %%  %%  %%      %%  %%        %%

        ======================================================
        ||                                                  ||
        ||        Kaixo ikasle11, Ongi etorri Arinara       ||
        ||         Hola ikasle11, Bienvenido a Arina        ||
        ||           Hi ikasle11, Welcome to Arina          ||
        ||                                                  ||
        ======================================================



=== HOST ===
kalk2020

=== DISK USAGE ===
You are using:        0.3 GB.
Your group is using: 6.6 GB.
Info updated at night.
Above 3 GB of disk usage it is charged. Rates: http://www.ehu.es/sgi/tarifas


Cluster Administrators:
Joaquim Jornet Somoza
Luca Bergamini

For any doubt or question contact us at:
izo-sgi@ehu.eus


[ikasle11@kalk2020 ~]$
```

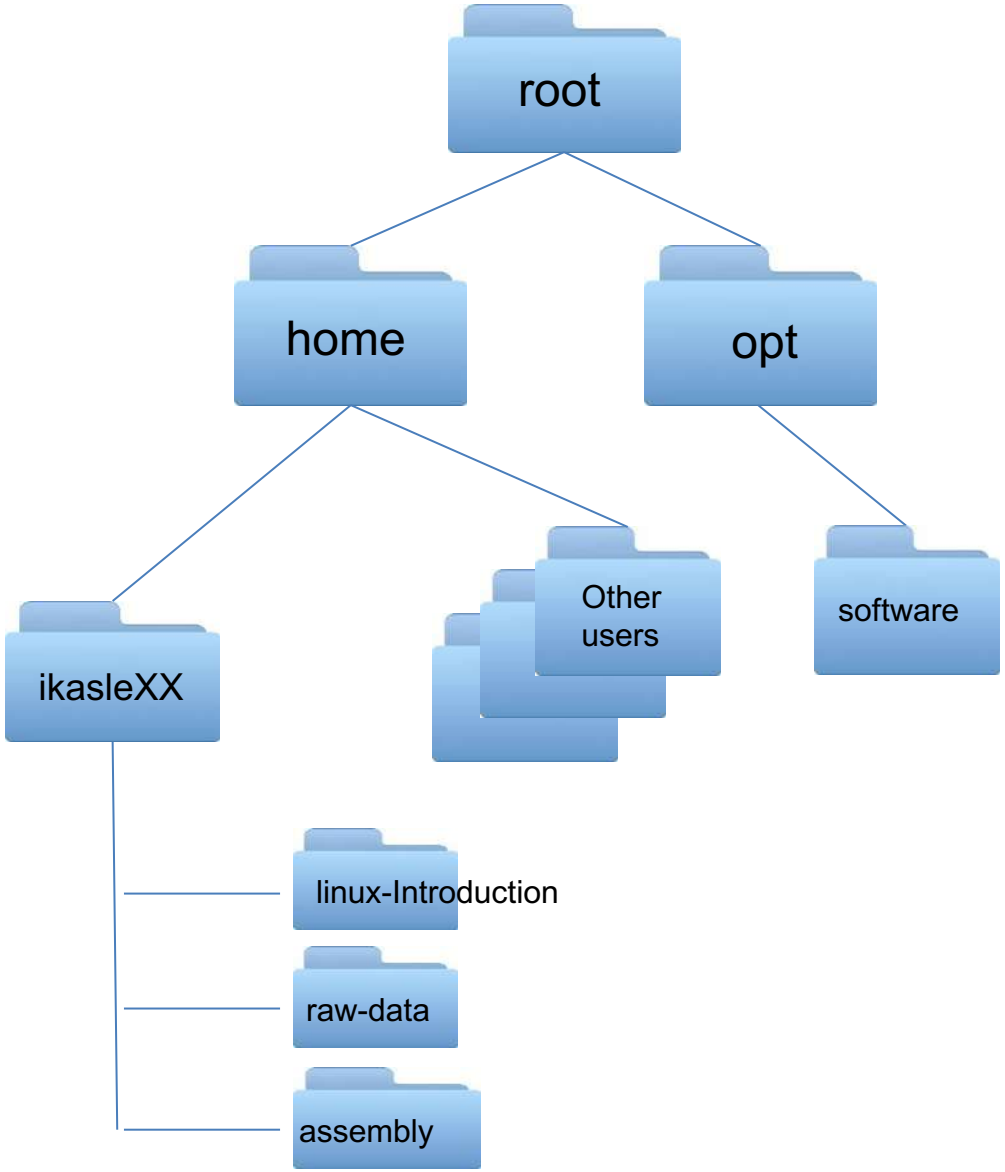
With the support of the Erasmus+ Programme of the European Union
2020-2026

GNU/Linux

ssh connexion

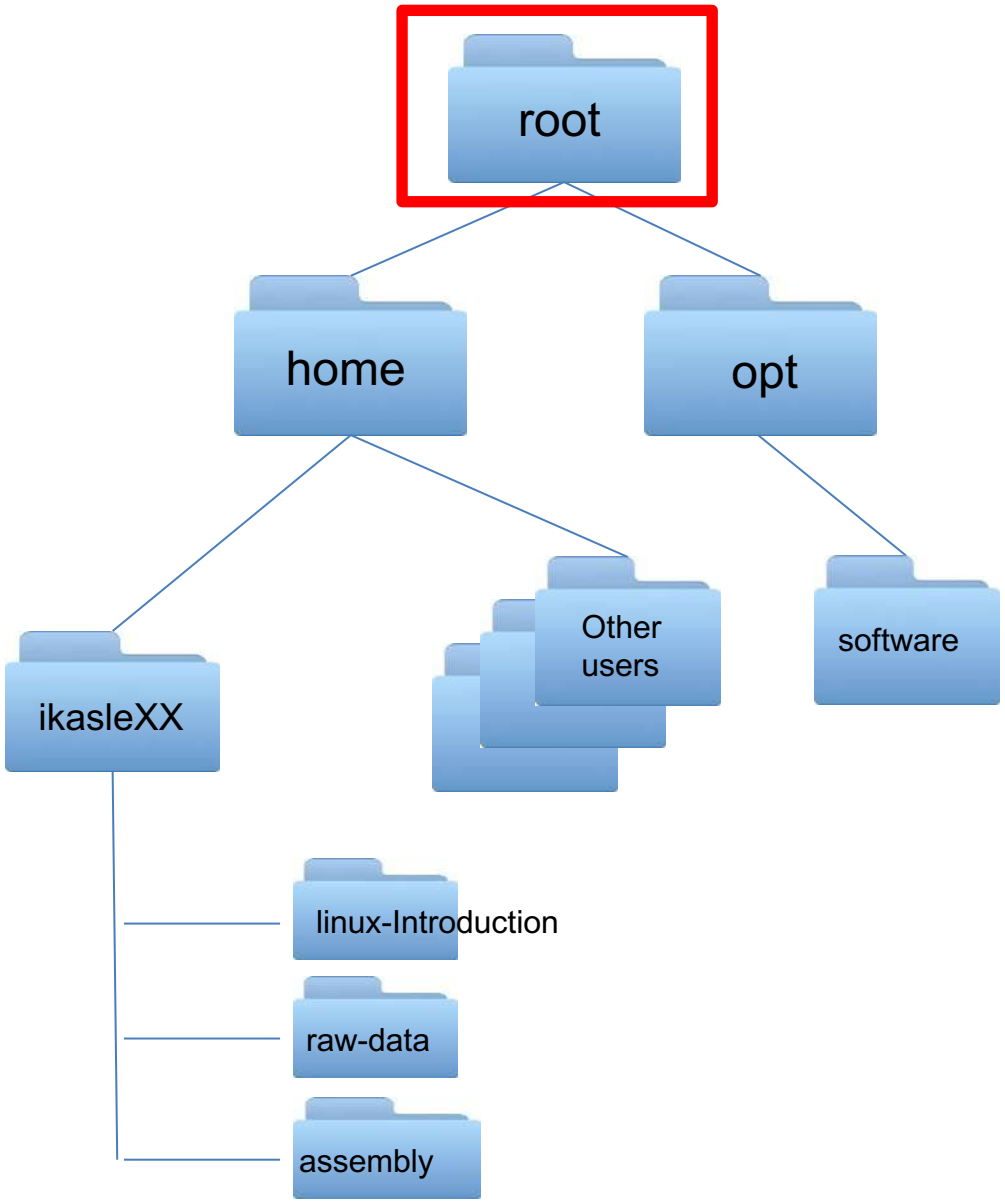
```
$ ssh -Y ikasleXX@kalk2020.lgp.ehu.es
$ password:

        %       %%%%%%    %%  %%%    %%        %
      %%%       %%    %%  %%  %%%%%  %%      %%%
     %% %%      %%    %%  %%  %%  %%%%%     %% %%
    %%   %%     %%%%%%    %%  %%     %%%    %%   %%
   %%%%%%%%%%   %%    %%  %%  %%     %%   %%%%%%%%%%
  %%        %%  %%     %%  %%  %%     %%  %%        %%

 ====================================================
 ||                                                ||
 ||        Kaixo ikasle11, Ongi etorri Arinara     ||
 ||         Hola ikasle11, Bienvenido a Arina      ||
 ||           Hi ikasle11, Welcome to Arina        ||
 ||                                                ||
 ====================================================

$ interactive -n 4 -p vfast -r mer1
$
```

GNU/Linux


root
home
opt
ikasleXX
Other users
software
linux-Introduction
raw-data
assembly

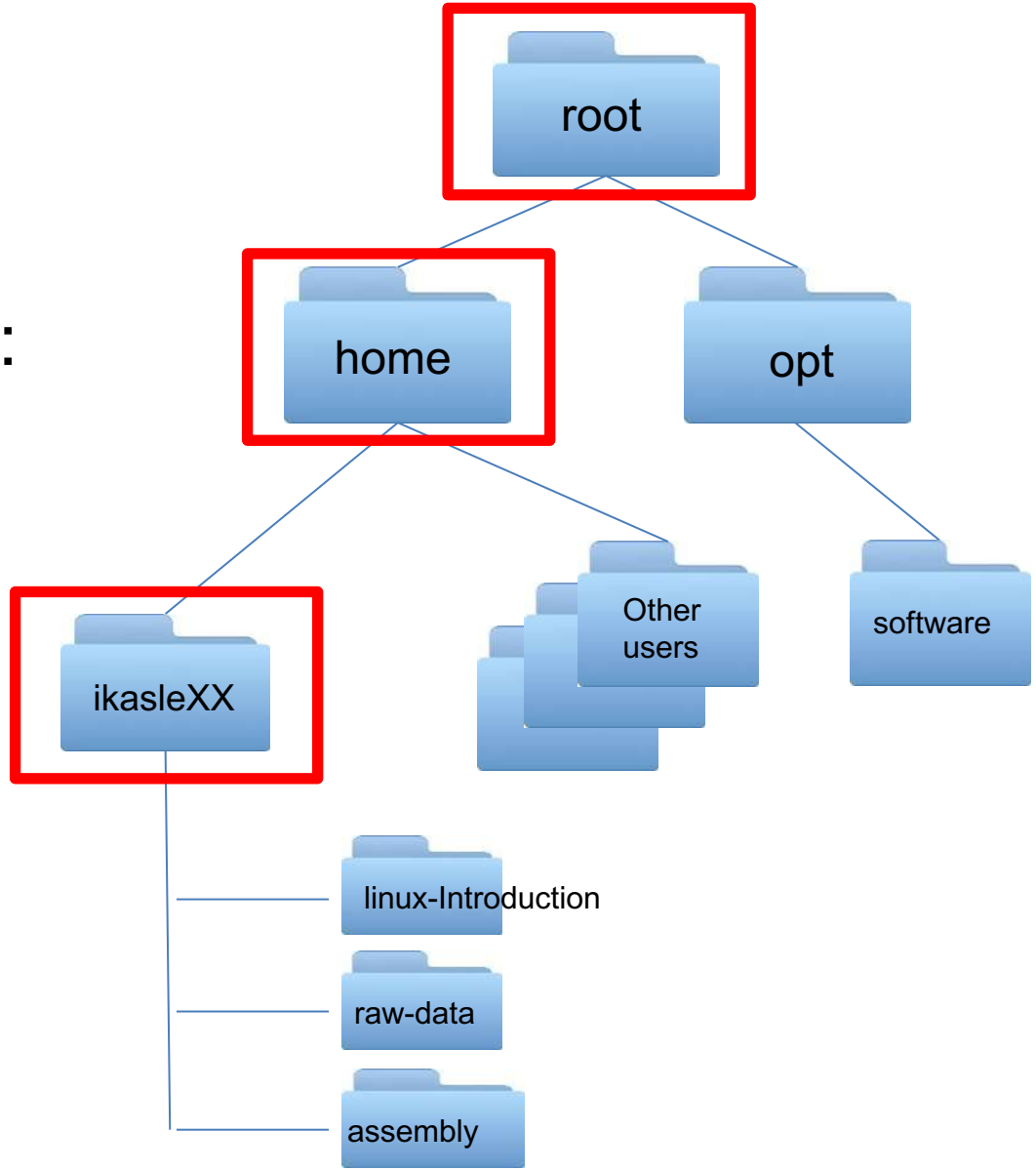
GNU/Linux

Path of root directory:
/



GNU/Linux

Path of your home-directory:
/home/ikasleXX/
or:
~



With the support of the Erasmus+ Programme of the European Union
2020-2026

GNU/Linux

Main commands

Linux cheat sheet

Cheatography

Linux Command Line Cheat Sheet
by Dave Child (DaveChild) via cheatography.com/1/cs/49/

Bash Commands

uname -a	Show system and kernel
head -n1 /etc/issue	Show distribution
mount	Show mounted filesy-stems
date	Show system date
uptime	Show uptime
whoami	Show your username
man *command*	Show manual for *command*

Bash Shortcuts

CTRL-c	Stop current command
CTRL-z	Sleep program
CTRL-a	Go to start of line
CTRL-e	Go to end of line

Bash Variables (cont)

export *NAME=value*	Set *$NAME* to *value*
$PATH	Executable search path
$HOME	Home directory
$SHELL	Current shell

IO Redirection

cmd < file
> Input of *cmd* from *file*

cmd1 <(cmd2)
> Output of *cmd2* as file input to *cmd1*

cmd > file
> Standard output (stdout) of *cmd* to *file*

cmd > /dev/null
> Discard stdout of *cmd*

Command Lists

cmd1 ; cmd2
> Run *cmd1* then *cmd2*

cmd1 && cmd2
> Run *cmd2* if *cmd1* is successful

cmd1 || cmd2
> Run *cmd2* if *cmd1* is not successful

cmd &
> Run *cmd* in a subshell

Directory Operations

pwd	Show current directory
mkdir *dir*	Make directory *dir*
cd *dir*	Change directory to *dir*
cd ..	Go up a directory
ls	List files

With the support of the Erasmus+ Programme of the European Union

GNU/Linux

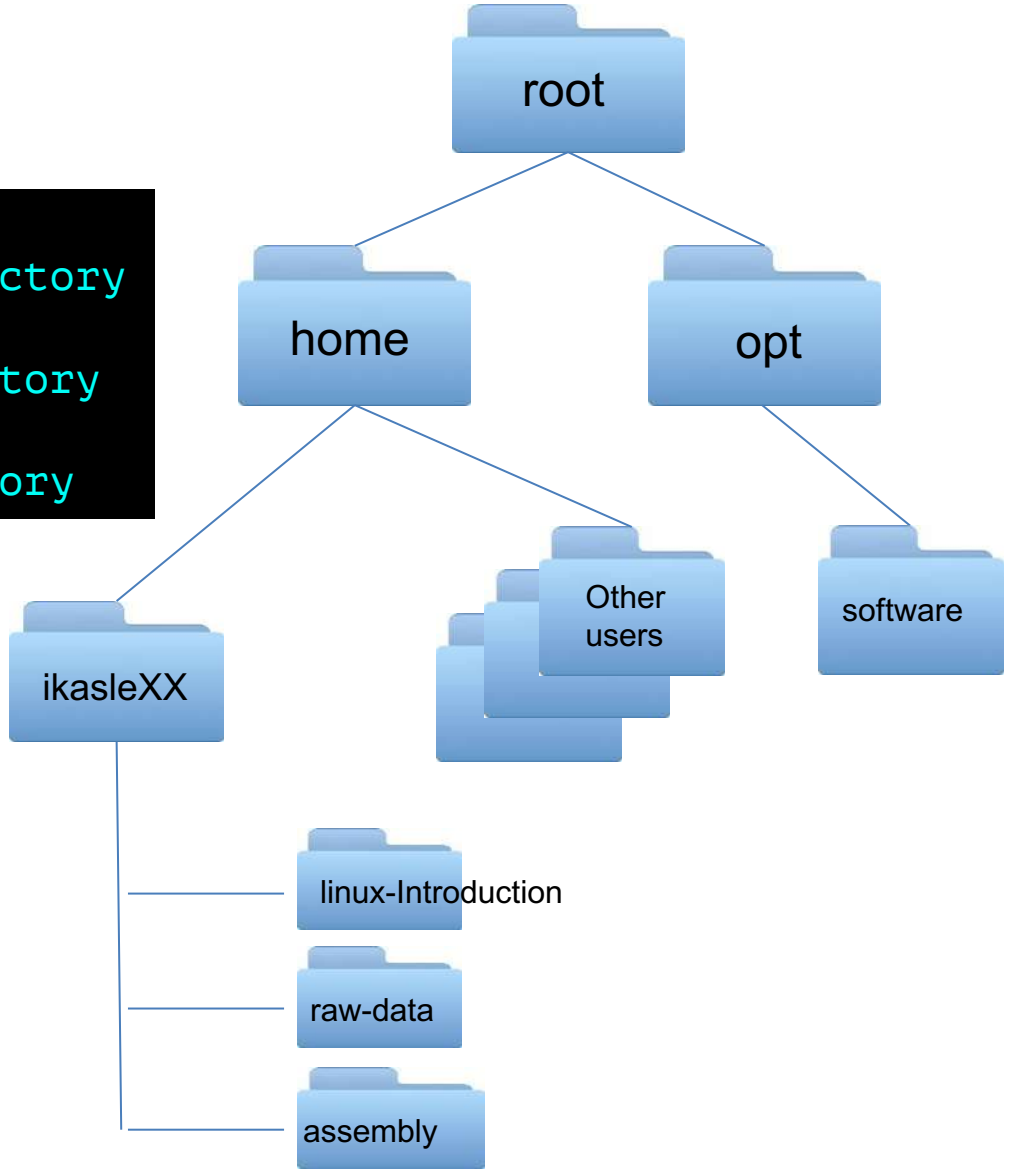
Main commands

```
$ cd
$ cd ../
$ cd -
$ mkdir
$ w
$ history
$ top
$ df
$ mv
$ vim
$ less example.txt
$ ln -s
```

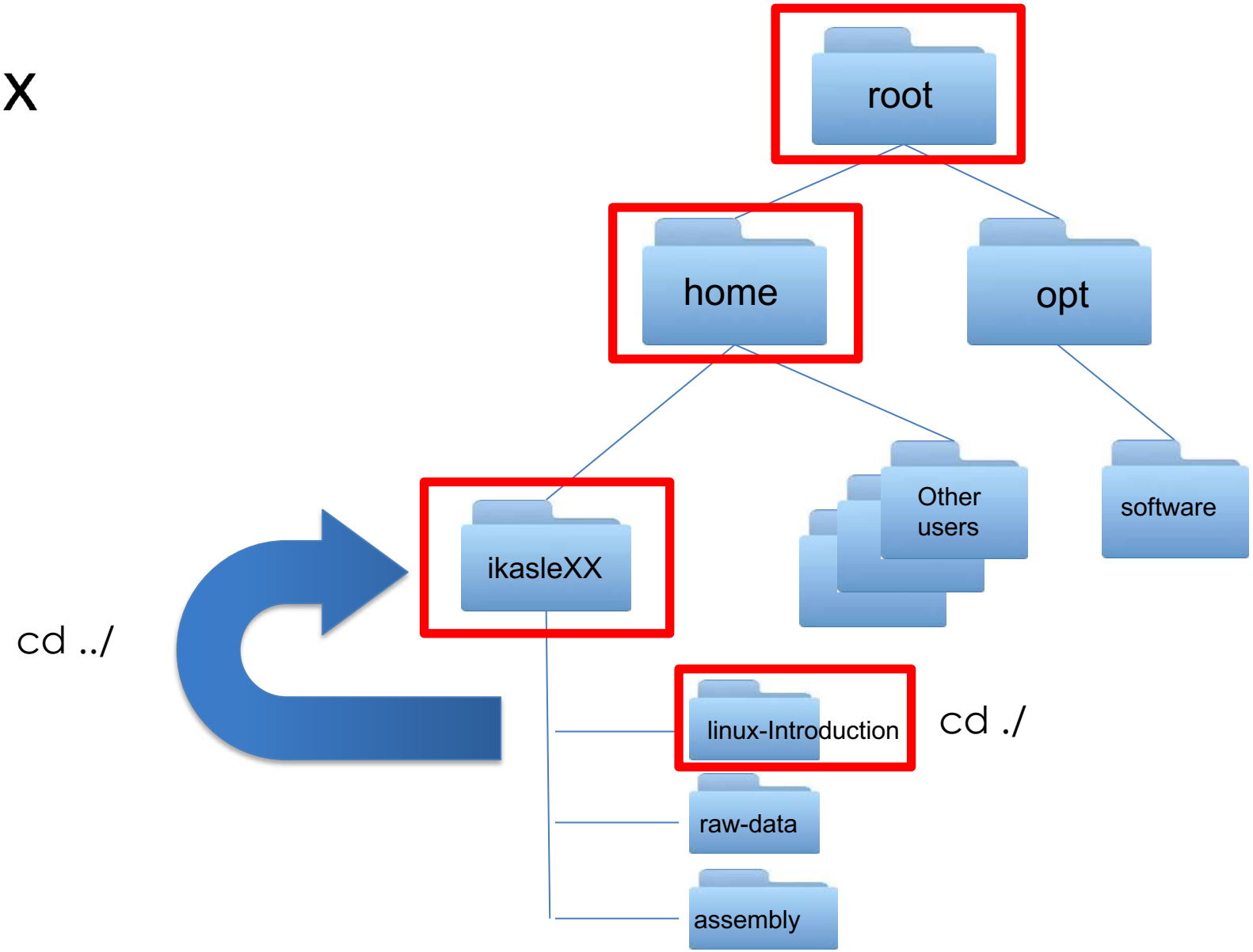
With the support of the Erasmus+ Programme of the European Union
2020-2026

GNU/Linux

```
# Abbreviation      # Meaning
cd . or cd ./       # Current working directory
cd .. or cd ../     # Next upper directory
cd -                # Last "visited" directory
cd ~                # Home directory
pwd                 # print working directory
```



GNU/Linux

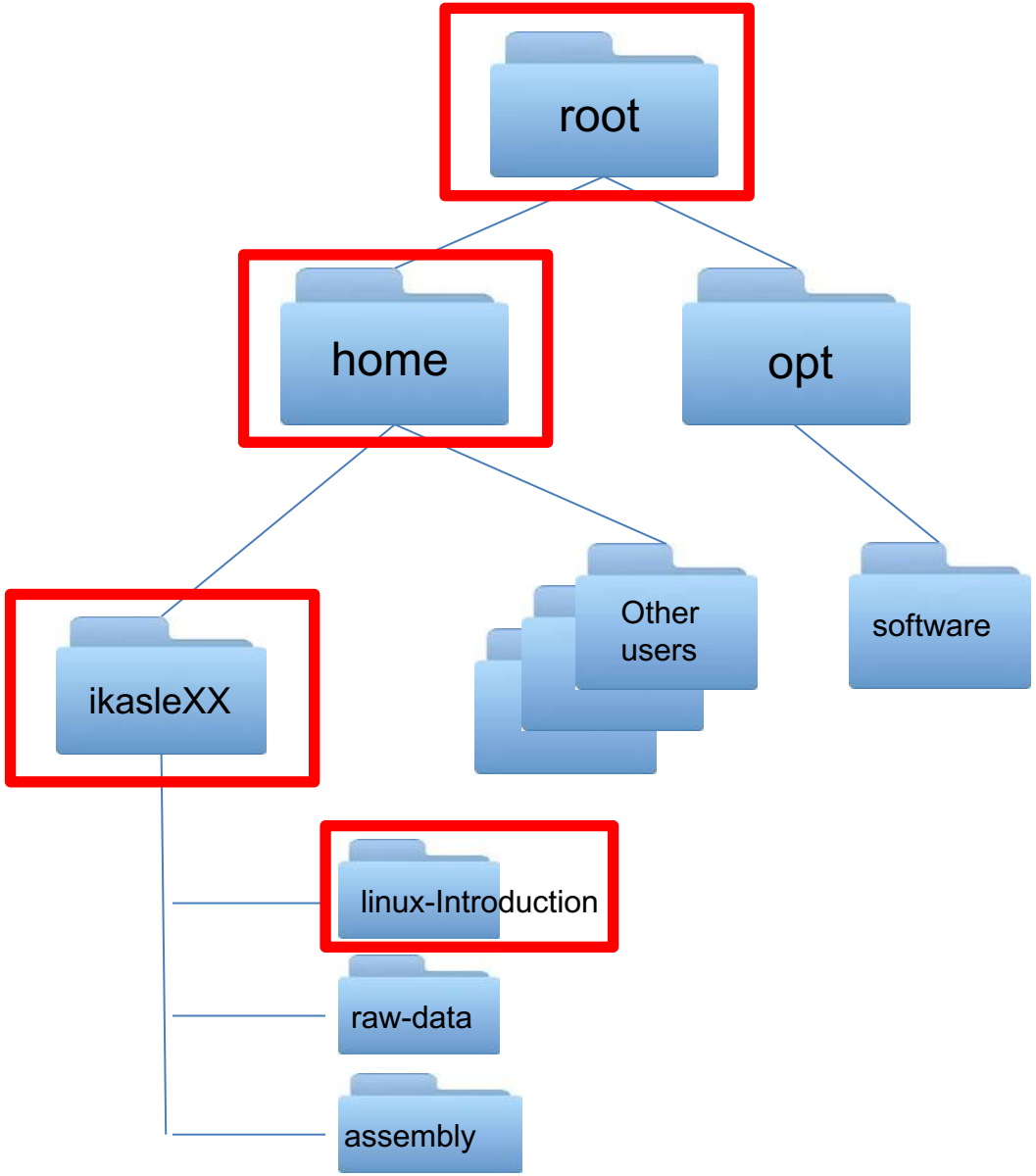

root
home
opt
ikasleXX
Other users
software
cd ../
linux-Introduction
cd ./
raw-data
assembly

GNU/Linux

Working directory:
/home/ikasleXX/linux-Intro

ABSOLUTE PATH:
/home/ ikasleXX/linux-Intro

RELATIVE PATH:
../ikasleXX/linux-Intro



GNU/Linux

Paths

- **Absolute path**: The absolute path of a file always starts at the root directory represented by "/" :

  /home/studentx/linux_introduction/exercise

- **Relative path**: The relative path of a file describes its location always relative to where the user is right now:

  ./linux_introduction/exercise

Your first commands!

Fundamental structure:

command [options] [argument]

Example 1:

ls

Example 2:

ls -sh

Example 3:

ls -sh ../gscratch

GNU/Linux

Your first commands!

Fundamental structure:

command [options] [argument]

Example 1:

```
$ wc
```

Example 2:

```
$ wc -l
```

Example 3:

```
$ wc -l filename.txt
```

GNU/Linux

Exercise 1:

1. Open your terminal
2. What‘s the path to your current directory?
3. How many folders does this directory contain?
4. Navigate to /home/username/ehu_genomic_data
5. Navigate to your home directory
6. Make a new directory called „linux_intro“
7. Navigate into the „linux_intro“ directory

With the support of the Erasmus+ Programme of the European Union

GNU/Linux

File handing commands

Command	Use
cp [file1] [file2]	Copy file1 and call it file2
mv [file1] [file2]	Rename file1 to file2
mv [file] /path/to/directory	Move file to directory
rm [file]	remove file
rmdir [directory]	remove directory
cat [file]	display file
more [file] , less [file]	display a file a page at a time
head [file]	display first 10 lines of a file
tail [file]	display last 10 lines of a file
grep “[keyword]“ [file]	display all lines in file containing the keyword
wc [file]	count number of lines/words/characters in file

File naming conventions

Why “uebung” rather than “übung”?
Why “ron_aniejo” rather than “ron_Añejo”?
Why “testfile” rather than “Testfile”?

Fundamentals:

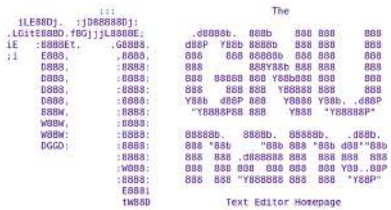
- **Avoid** characters such as **/ * & %**
- **NO WHITE SPACES!**
- **ONLY** use letters and numbers and replace whitespaces with the following special characters: _ . + -

e.g. chironomus-riparius+piger.fasta
thats_my_logfile.txt
thatsMyLogfile.txt

GNU/Linux

Create files

Example of built-in text editors on Unix command line:



- nano

- vim

GNU/Linux

Create files

Create files in a plain text editor (not Word):

Notepad++

https://notepad-plus-plus.org/

Most popular more advanced code editors

Visual Studio Code

ATOM

Sublime Text

https://code.visualstudio.com https://atom.io https://www.sublimetext.com

GNU/Linux

Files inspection

See the whole file

$ *cat <file>*

Inspecting the file

$ *less <file>* # Within less search for an specific word typing /"word"

Within less, exit typing q

Seeing the top of the file

$ *head <file>* # head [- number] <file>

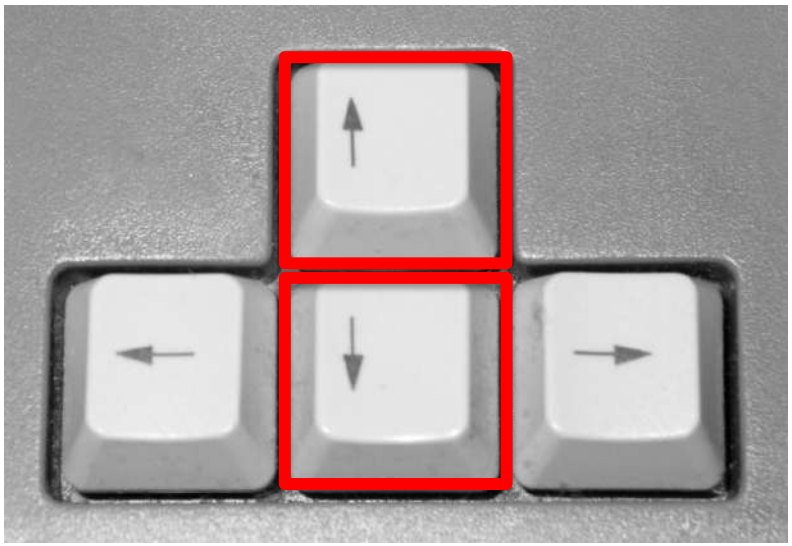
Seeing the end of the file

$ *tail <file>* # tail [- number] <file>

GNU/Linux

- Used commands → history

#gives you all the commands previously run in your session



GNU/Linux

Autofill

What happens if you write/press ***head Te*[tab]**?

GNU/Linux

Deleting files and directories

Attention!!

The **system** always **assumes** the **user knows** what she/he is doing

- No query whether you **REALLY** want to delete something …
- No recycle bin – **Deleted means Deleted**
- Particular caution needed when using wildcards

GNU/Linux

Exercise 2:

1. Create a textfile called „delete_me.txt“ using *touch [filename]*.
2. Create a directory called „delete_me“.
3. Move „delete_me.txt“ to „delete_me“.
4. Delete the directory

GNU/Linux

File permissions

```
ikasle01@kalk2020:~/gscratch/TEST2$ ll
total 9
drwxr-xr-x 1 ikasle01 pomaster 24 Nov 23 15:03 File.txt
```

drwxr-xr-x

Type User Group Others

r	read
w	write
x	execute
-	denied

GNU/Linux

Save output

Instead of printing the output to the terminal, one can save it in a new file

Command (options) file > new_file

For example:

head inputfile.txt > outputfile.txt

GNU/Linux

```
Exercise 3:

1) Save the first line of the file
   „WatsonCrick1953.txt“ as „papertitle.txt“

2) Save the remaining lines as „mainbody.txt“.
```

GNU/Linux

Piping commands

Unix offers the possibility to pipe commands. By doing this, the output of the first command serves as the input of the following one

How many sequences does twogenes.fa contain?

grep ">" twogenes.fa | wc –l

GNU/Linux

Piping commands

| head – check the output

“ | head“ offers the possibility to check the command/ouput

Avoids (unnecessarily) huge amounts of output files

Especially useful when working on huge files or using complex (piped) commands

GNU/Linux

Listing 1. Example of cat to create a file:

```
ikasle01@kalk2020:~$ cat > species.list
snail
octopus
mussel
wolf
<ctrl d>
```

GNU/Linux

Listing 2. Example of cat to append a file:

```
ikasle01@kalk2020:~$ cat >> species.list
wasp
jellyfish
<ctrl d>
```

GNU/Linux

Listing 3. Example of cat without flags:

```
ikasle01@kalk2020:~$ cat species.list
snail
octopus
mussel
wolf
wasp
jellyfish
ikasle01@kalk2020:~$
```

GNU/Linux

Listing 4. Example of cat to count lines:

```
ikasle01@kalk2020:~$ cat -n species.list
     1    snail
     2    octopus
     3    mussel
     4    wolf
     5    wasp
     6    jellyfish
ikasle01@kalk2020:~$
```

Use of 'wc'

The wc (wordcount) command counts the number of lines, words (separated by whitespace), and byte count in specified files, or from stdin.

```
ikasle01@kalk2020:~$ wc species.list
6  6 41 species.list
ikasle01@kalk2020:~$ wc -l species.list
6 species.list
ikasle01@kalk2020:~$ wc -w species.list
6 species.list
ikasle01@kalk2020:~$ wc -c species.list
41 species.list
```

GNU/Linux

paste

```
$ paste species.list2 species.list > species.list3
$ cat species.list3
Gastropoda      snail
Cephalopoda     octopus
Bivalvia        mussel
Carnivora       wolf
Hymenoptera     wasp
Cnidaria        jellyfish
```

Use of ‘grep’

The grep command searches specified files or stdin for patterns matching a given expression(s). Output from grep is controlled by various option flags.

```
ikasle01@kalk2020:~$ grep Carnivora species.list3 species.list4
species.list3:Carnivora wolf
species.list4:Carnivora wolf
species.list4:Carnivora Tiger
species.list4:Fox Carnivora
species.list4:Carnivoras seal
```

GNU/Linux

Example of grep – counting number of matches in files:

```
ikasle01@kalk2020:~$ grep -c Carnivora species.list3 species.list4
species.list3:1
species.list4:4
```

GNU/Linux

Example of grep – suppress filename in output:

```
ikasle01@kalk2020:~$ grep -h Carnivora species.list3 species.list4
Carnivora   wolf
Carnivora   wolf
Carnivora   Tiger
Fox   Carnivora
Carnivoras seal
```

GNU/Linux

Example of grep – case insensitive:

```
ikasle01@kalk2020:~$ grep -i Carnivora species.list3 species.list4
Carnivora   wolf
Carnivora   wolf
Carnivora   Tiger
Fox   Carnivora
carnivora   dingo
carnivoras  bear
Carnivoras  seal
```

With the support of the Erasmus+ Programme of the European Union
2020-2026

GNU/Linux

Example of grep – inverted matching:

```
ikasle01@kalk2020:~$ grep -v Carnivora species.list3 species.list4
species.list3:Gastropoda      snail
species.list3:Cephalopoda     octopus
species.list3:Bivalvia  mussel
species.list3:Hymenoptera     wasp
species.list3:Cnidaria  jellyfish
species.list4:Gastropoda      snail
species.list4:Cephalopoda     octopus
species.list4:Bivalvia  mussel
species.list4:Hymenoptera     wasp
species.list4:Cnidaria  jellyfish
species.list4:Gastropoda      slug
species.list4:Helicoidea      Snail
species.list4:carnivora dingo
species.list4:carnivoras      bear
```

GNU/Linux

Example of grep – combining flags:

```
ikasle01@kalk2020:~$ grep -v -i Carnivora species.list3 species.list4
species.list3:Gastropoda  snail
species.list3:Cephalopoda octopus
species.list3:Bivalvia    mussel
species.list3:Hymenoptera wasp
species.list3:Cnidaria    jellyfish
species.list4:Gastropoda  snail
species.list4:Cephalopoda octopus
species.list4:Bivalvia    mussel
species.list4:Hymenoptera wasp
species.list4:Cnidaria    jellyfish
species.list4:Gastropoda  slug
species.list4:Helicoidea  Snail
```

GNU/Linux

Example of grep – word matching:

```
ikasle01@kalk2020:~$ grep -w Carnivora species.list3 species.list4
species.list3:Carnivora wolf
species.list4:Carnivora wolf
species.list4: Carnivora      Tiger
species.list4: Fox      Carnivora
```

GNU/Linux

Use of ‘awk’

Actual programming language

Input file is not changed

Especially useful when working with tables

Field-oriented

Use of ‘awk’

```
ikasle01@kalk2020:~$ awk '{print$1}' species.list4 | head
Gastropoda
Cephalopoda
Bivalvia
Carnivora
Hymenoptera
Cnidaria
Gastropoda
Helicoidea
Carnivora
Fox
```

Use of ‘awk’

```
ikasle01@kalk2020:~$ awk '{print$2}' species.list4 | head
snail
octopus
mussel
wolf
wasp
jellyfish
slug
Snail
Tiger
Carnivora
```

Use of ‘awk’

```
ikasle01@kalk2020:~$ awk '{print$2,$1}' species.list4 | head
snail Gastropoda
octopus Cephalopoda
mussel Bivalvia
wolf Carnivora
wasp Hymenoptera
jellyfish Cnidaria
slug Gastropoda
Snail Helicoidea
Tiger Carnivora
Carnivora Fox
```

GNU/Linux

Other interesting commands

cut

Cuts parts of files based on certain criteria

cut –f [column_number]

cut –c [number_of_characters]

GNU/Linux

Other interesting commands

sort

Sorts files according to set criteria

sort –k [beginning of Column, end of Column]

sort –n

sort –u

GNU/Linux

Other interesting commands

uniq

- Uniq combines identical neighboring lines into one

Example:

original		After uniq
1		1
1	uniq →	2
2		3
3		4
3		
3		
4		

GNU/Linux

Other interesting commands

sed

- **S**tream **ED**itor
- Actual (simple and compact) programming language
- Parses and transforms text
- Line-oriented
- Input file is not changed
- Most famous application: text substitution

GNU/Linux

Other interesting commands

sed

substitution

sed ‘s/original/substitution/‘ [file]

GNU/Linux

loops

'for' loop

```
for variable in list
do
# commands to be executed for each item in the list
done
```

GNU/Linux

loops

‘for’ loop

```
for i in seq 5
do
        echo Wellcome $i times
done
```

GNU/Linux

loops

'for' loop

```
for i in seq 5
do
        echo Wellcome $i times
done
```

```
for i in $(seq 5); do echo Wellcome $i times; done
```

GNU/Linux

loops

'while' loop

while [condition]
do
commands **to** be executed **while** the condition **is true**
done

GNU/Linux

loops

'while' loop

```
count=1
while [ $count –le 5 ]
do
        echo $count
        count=$((count + 1))
done
```

```
count=1; while [ $count -le 5 ]; do echo $count $(( count++ )); done
```

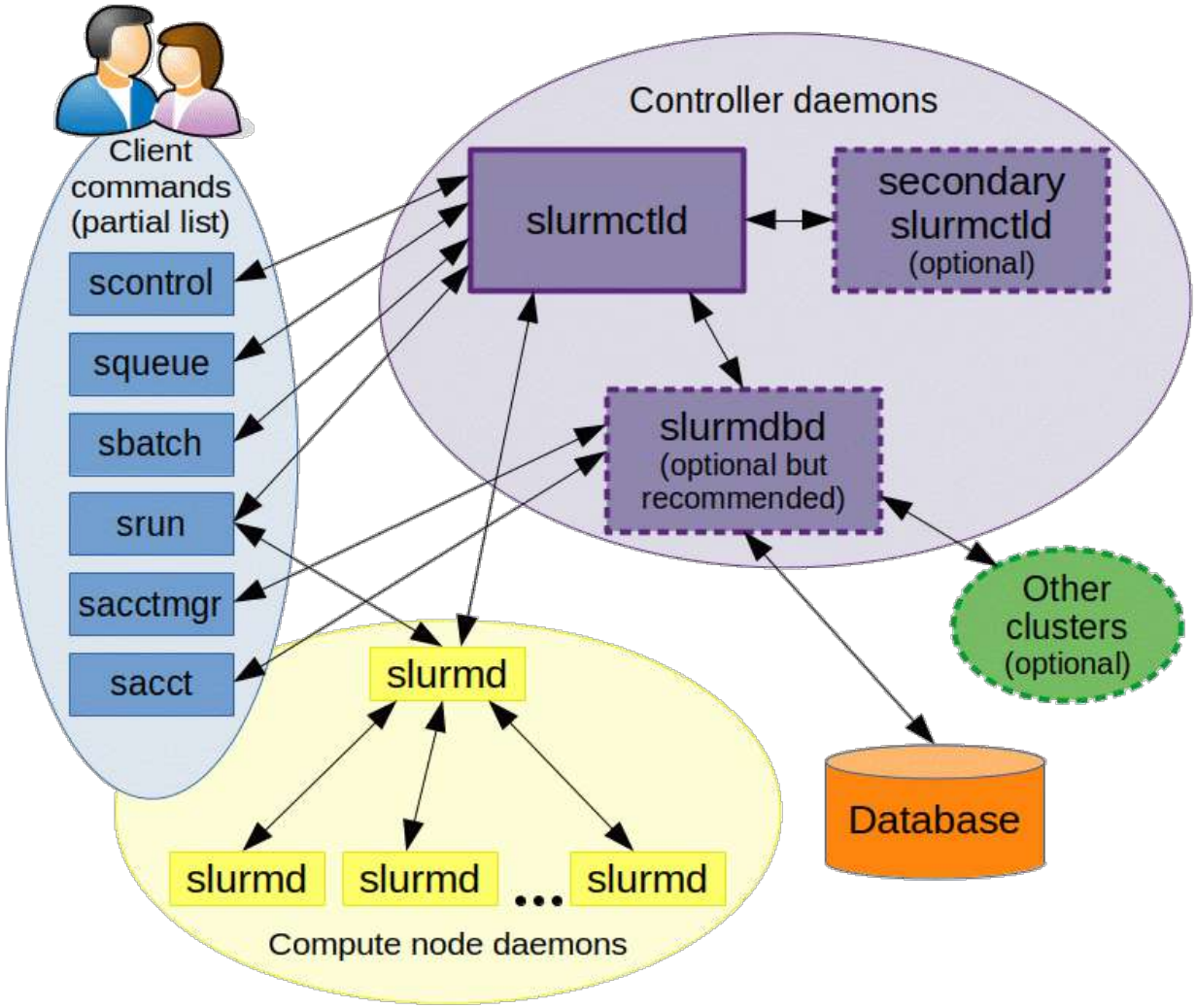
Things to remember

- Document all steps (will save you time on the long run)
- Save all your commands in one, clearly structured folder
- Document all errors and their solutions
- Search online for error solutions; likely someone else experienced the same problem before
- 99.99% of the errors are user errors
- No error does not mean the output is useful
- Always check your output files (file size and content)
- Find the right balance between understanding the software and accepting you can not know everything

slurm
workload manager

Simple Linux Utility for Resource Management

slurm
workload manager



sinfo

reports the state of partitions and nodes managed by **Slurm**. It has a wide variety of filtering, sorting, and formatting options

```
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
main*        up   infinite      1  alloc caius
main*        up   infinite      5   idle
augustus,caesar,claudius,nero,tiberius
head         up   infinite      1   idle claudius
test         up   infinite      3   idle test[1-3]
tbg          up   infinite      3   idle test[1-3]
gpu          up   infinite      1   idle jupiter
```

srun

is used to submit a job for execution in real time.

```
$ srun -N 2 -n 48 -t 30 -A xz0123 ./my_small_test_job
```

sbatch

is used to submit a job script for later execution. The script will typically contain one or more **srun** commands to launch parallel tasks.

```
$ sbatch example.sh
Submitted batch job 119405
```

example.sh

```
#!/bin/bash
#
#SBATCH -p generic              # Partición (cola)
#SBATCH -N 1                    # Numero de nodos
#SBATCH -n 1                    # Numero de cores (CPUs)
#SBATCH -mem 100                # Bloque de memoria para todos los nodos
#SBATCH -t 0-02:00              # Duración (D-HH:MM)
#SBATCH -o slurm.%N, %j.out     #STDOUT
#SBATCH -e slurm.%N, %j.err     #STDERR
#SBATCH --mail-type=END,FAIL    # Notificación cuando el trabajo termina o
falla

#SBATCH -- mail-user=micorreo@ehu.eus # Enviar correo a la dirección
```

scancel

is used to cancel a pending or running job or job step. It can also be used to send an arbitrary signal to all processes associated with a running job or job step.

```
$ scancel 119405
```