

# Introduction to Linux and the Shell:

## Useful tools and commands for bioinformatic analyses

---

Angelika Merkel (Head of Bioinformatics Unit IJC)  
03/06/2024

# The IJC Bioinformatics Unit



Angelika Merkel  
Unit Manager  
Bioinformatician



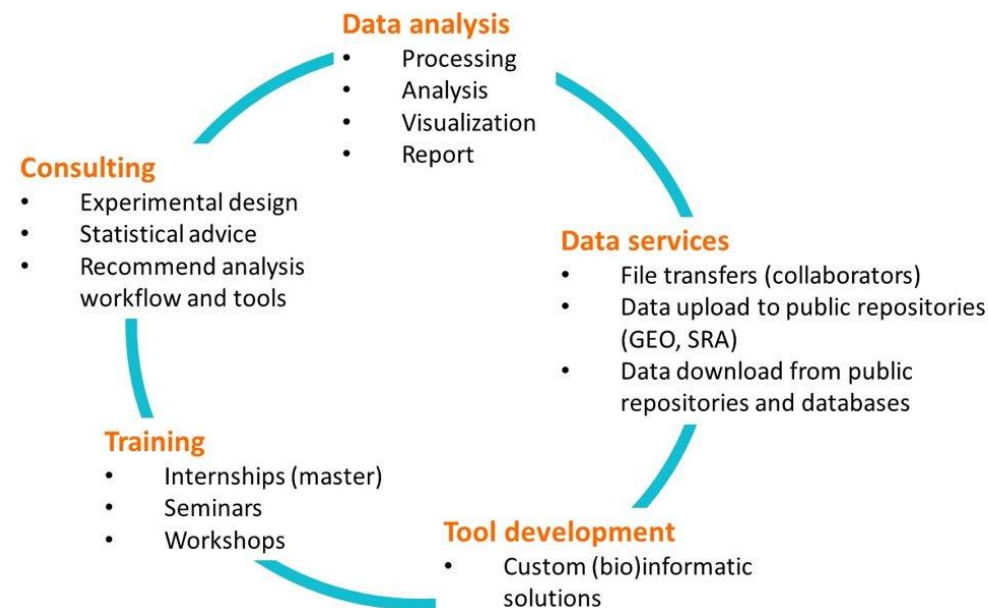
Emilio Lario  
Software Engineer



Marina Vilardell  
Bioinformatician



Marta Meroño  
Bioinformatician  
(Student)



Office: 1st floor; phone: 4300

<https://ijcbit.eu>

<https://www.carrerasresearch.org/en/bioinformatics-unit>



# Workshop overview

---

## Day 1:

- Introduction to Linux
- Practical session I: Linux & the Shell (basics and commands)
  - Moving around the file system
  - File properties
  - Basic file and directory handling
  - Text file viewing and manipulation

## Day 2:

- Practical session II: Shell programming & vim (CLI text editor)
  - Create and run a shell script
  - Bash variables and
  - Control structures (if-else, for-loop, while-loop)

## Workshop overview

---

Course materials:

<https://ijcbit.github.io/Workshops/>

Course server ('Linux course'):

<https://vpn.carrerasresearch.org>

# What is Linux?

---

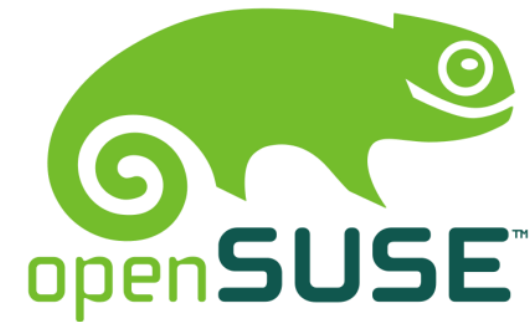
= free-open source computer software environment (operating system)

Free Software license:

- Freedom to run the program for any purpose
- Freedom to study and change the program; access to underlying source code
- Freedom to share copies to help your neighbor
- Freedom to distribute copies of modified versions for others

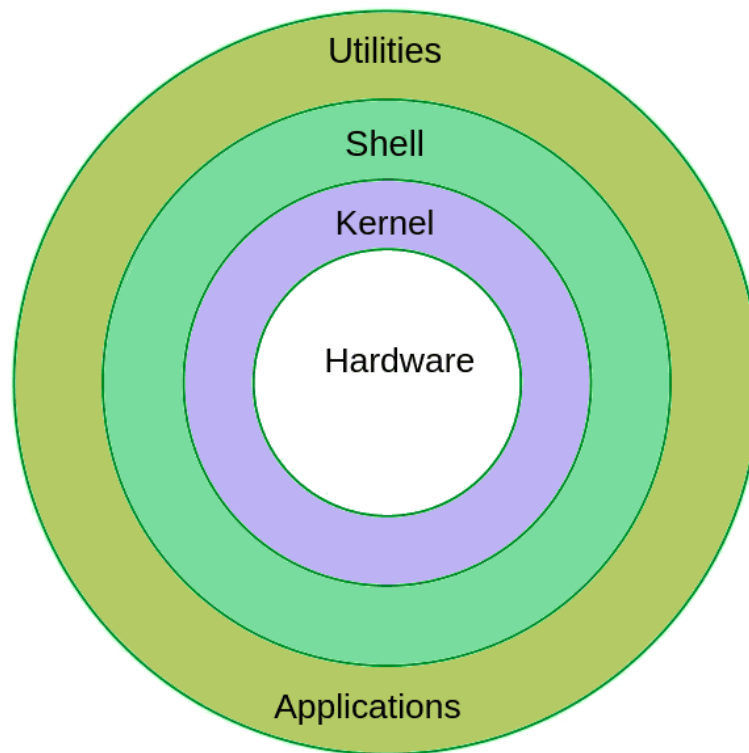
# More than 300 Linux distributions!

---



[https://upload.wikimedia.org/wikipedia/commons/6/6f/Linux\\_distros\\_tree.png](https://upload.wikimedia.org/wikipedia/commons/6/6f/Linux_distros_tree.png)

# The Linux system



outer



core

---

Utilities & applications

programs

X

graphical system that provides windows, menus, icons, mouse support (KDE, GNOME)

Shell

user interface for typing commands, executing them and displaying the results

---

Kernel

low-level operating system for handling files, disks, networking, etc.

---

<https://www.instructables.com/Linux-Presentation-in-PDF/>

# Terminology

---

**Shell** = a command line interface: CLI (as supposed to graphical user interface: GUI)  
multiple shells: *bsh*, *bash*, *zsh*

**Bash** = Bourne Again SHell (enhanced version of the original Unix shell program,  
bsh written by Steve Bourne)

**Terminal** = a program called *terminal emulator*, opens a window and lets you interact  
with the shell (konsole, xterm, gnome-terminal, etc)



# Let's get started..

---

Connect to the course server:

1. Open web browser  
`https://vpn.carrerasresearch.org/ > Login >> Linux_course`
2. Open terminal  
`ssh username@intercept`

# First steps

---

1. The prompt  
username@host:currentdirectory\$
2. A typical command *ls*  
program [options] arguments  
ls # list files  
ls --help # help option  
ls -a # list all files (including)  
ls -a mydirectory # list all files in mydirectory  
ls -l # use a long listing format
3. File types:
  - directories (blue),
  - regular files (white),
  - executables (green),
  - compressed files (red),
  - softlinks (lightblue)      \*colors depend on bash configuration

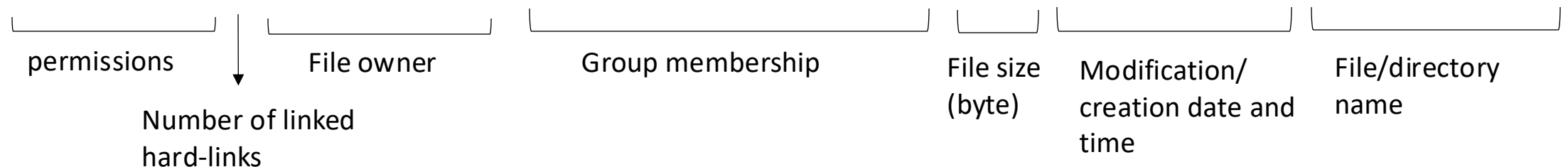
# File attributes

ls: list information about files, default current directory

- l # use a long listing format
- h, --human-readable # with -l and -s, print sizes like 1K 234M 2G etc.
- t # sort by modification time, newest first

```

amerkel@INTERCEPT:/home$ ls -lth
total 44K
drwx----- 6 amerkel      isilon_merkel_group 4,0K nov 28 11:04 amerkel
drwxrwxrwt  2 super       root                4,0K nov 28 10:39 shared
drwx----- 5 idevillasante isilon_merkel_group 4,0K nov 28 10:10 idevillasante
drwx----- 5 jalcantara    isilon_admins       4,0K nov 28 09:41 jalcantara
drwxr-xr-x 19 super       super                4,0K nov 28 09:40 super
  
```



# File permissions

```

amerkel@INTERCEPT:/home$ ls -lth
total 44K
drwx----- 6 amerkel      isilon_merkel_group    4,0K nov 28 11:04 amerkel
drwxrwxrwt  2 super       root                   4,0K nov 28 10:39 shared
drwx----- 5 idevillasante isilon_merkel_group    4,0K nov 28 10:10 idevillasante
drwx----- 5 jalcantara    isilon_admins          4,0K nov 28 09:41 jalcantara
drwxr-xr-x 19 super       super                  4,0K nov 28 09:40 super

```

☐ ☐ ☐

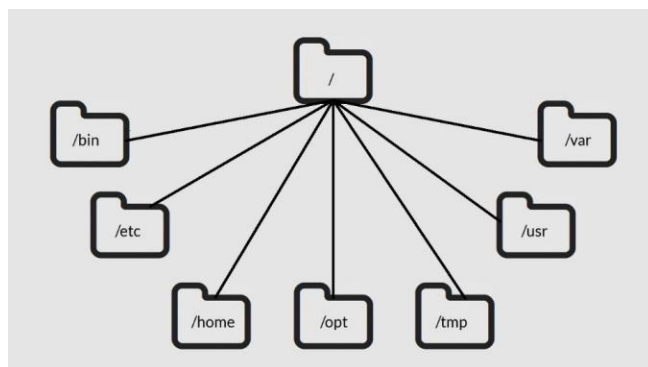


o,g,a      = owner, group, all  
r,w,x      = read, write, execute    # basic file permissions

chmod                      # change permissions of a file or directory  
\$ chmod a+rx myfile        # add read and execute permissions for all for file

# Linux directory structure

## Tree hierarchy



Tip: Display directory structure with 'tree' command

Dir	Description
/	The directory called "root." It is the starting point for the file system hierarchy. Note that this is not related to the root, or superuser, account.
/bin	Binaries and other executable programs.
/etc	System configuration files.
/home	Home directories.
/opt	Optional or third party software.
/tmp	Temporary space, typically cleared on reboot.
/usr	User related programs.
/var	Variable data, most notably log files.

\*E1

# Finding your way around

---

1. Where am I ?

```
pwd # print current working directory
```

2. What is a path ?

```
/directory/directory/directory
```

3. Change directory 'cd '

<code>cd somedirectory</code>	<code># go to some directory</code>
<code>cd . ;</code>	<code># go to current directory</code>
<code>cd ..;</code>	<code># go one level up</code>
<code>cd ../../</code>	<code># go two levels up</code>
<code>cd -</code>	<code># go to previous</code>
<code>cd [~]</code>	<code># go /home/username</code>

# Basic file and directory operations

---

<code>mkdir mydirectory</code>	<code># create a directory</code>
<code>cp file newfile</code>	<code># copy file</code>
<code>cp -r mydirectory newdirectory</code>	<code># copy directory (recursively)</code>
<code>mv filename newfilename</code>	<code># move (rename) file or directory to new</code>
<code>mv file directory</code>	<code># move file into directory</code>
<code>rm file</code>	<code># remove (delete) file</code>
<code>rm -r directory</code>	<code># remove (delete) directory (recursively)</code>

## Softlinks

<code>ln -s file -n softLink</code>	<code># create a soft link to a file</code>
<code>rm softLink</code>	<code># remove the softlink (not the original file)</code>


\*E2

# Handy short cuts

---

## 1. Command history:

history

use  to go back and forward

## 2. autocomplete filenames and commands

Wildcard substitution '\*' = substitute for anything

`ls *.bed` # list all filenames starting with anything and ending with '.bed'

`ls les*` # list all filenames starting with 'les' and ending with anything

\*E2



# File compression and archiving

---

## # Compression decreases file size

```
gzip, gunzip    # compress or uncompress files in GNU zip format (.gz)
bzip2, bunzip2  # compress or uncompress files in Burrows-Wheeler format (.bz2)
zip, unzip      # compress or uncompress file in Windows zip format (.zip)
```

```
*gzip myfile # produces myfile.gz and the original file is deleted
```

```
zcat myfile # uncompress to standard output
```

## # Archiving packs directories and files into a single package preserving hierarchy

```
tar -tf archive.tar          # list contents of archive
tar -xf archive.tar          # unpack archive
tar -cf archive.tar dir1 dir2 # create archive from directory 1 and directory 2
tar -xzf archive.tar.gz      # unpack and extract gzipped archive
```

\*E3

# File viewing and info

---

<code>cat myfile</code>	<code># print myfile content</code>
<code>less (more) myfile</code>	<code># view myfile by page (use spacebar)</code>
<code>head -10 myfile</code>	<code># view the first 10 lines of myfile</code>
<code>tail -10 myfile</code>	<code># view the last 10 lines of myfile</code>
<code>wc -cwl myfile</code>	<code># count characters, words, lines in myfile</code>

# Shell input/output and error

---

Default input device = keyboard, Default output device = screen

- Command output can be re-directed to a file:  
\$ mycommand > outfile # create/overwrite outfile  
\$ mycommand >> outfile # append output to outfile
- Standard error are system messages written standard output  
\$ ls lala  
ls: cannot access 'lala': No such file or directory
- Standard error can be directed to a file as well:  
\$ mycommand 1> outfile  
\$ mycommand 2> errorlog

# Merging files

---

```
Cat file1 file2 > newfile    # concatenate two file into a new file
Cat file1 >> newfile          # append a file to another

paste file1 file2            # combine two text files side by side
```

\*E4

# File text manipulations

---

```
Sort      # print lines of text file in specified order
          (default: alphabetical)
  -n      # sort numerically
  -r      # reverse order
  -k f    # sort by field f
  -c      # don't sort, only check if it is sorted
```

Example:

```
$ sort -k 5n myfile    # sort by 5th field numerically
```

# File text manipulations

---

Cut           # extract columns of a text file

    -f 1,3     # extract columns 1 and 3  
              (ranges: 1-3 = 1 to 3; 5- = 5 to last; -16 = first to 16 )

    -d,        # field separator (default: tab separator)

Uniq          # filter matching line from input

    -c         # count number of unique lines

    -u         # print only unique lines

    -d         # print only duplicated lines

\*E5

# The pipe '|' operator

---

Using the shell, you can redirect standard of one command to be the standard input of another:

```
$ cut -f1 peaks.bed | uniq -c | sort -nr
```

# File text manipulations with 'grep'

---

Grep # print all lines matching a regular expression

- v # print lines that do not match the regular expression
- w # match only complete words
- c # print a count of matching lines
- A N # after each matching line, print the next N lines from this file
- B N # before each matching line, print the next N lines from this file
- E # or egrep for extended regular expression

Example:

```
$ grep chr1 peaks.bed
```

\*E6



## Online tutorials Linux

---

<https://ryanstutorials.net/linuxtutorial/>

# Introduction to the Linux terminal

## Day 2: Programming with the Shell

---

Angelika Merkel (Head of Bioinformatics Unit IJC)

04/06/2024

# Vim command line text editor

```
VIM - Vi IMproved  
  
version 8.0.1763  
by Bram Moolenaar et al.  
Modified by <bugzilla@redhat.com>  
Vim is open source and freely distributable
```

```
Vim shellscript.sh    # opens a new file 'shellscript.sh' with vim
```

## Basic vim:

```
[esc]  # switch mode
i      # enter edit mode
:      # enter command mode

wq     # when in command modes: save and exit vim
q!    # when in command mode: exit without saving
```

For more on vim, check [here](#)

# Shell script

---

## 1. create shellscript.sh

```
#!/bin/bash

### Author:
### Date:
### Description:
### My first shell script

# do something
echo "Hello world!"
```

## 2. execute

```
# make the script executable
$ chmod a+x shellscript.sh

# run the script
./shellscript.sh
```

# Shell variables: \$

---

## Variables

```
MYVAR="Hello world!" # assign a variable
echo $MYVAR          # return variable
printf $MYVAR        # print variable formatted
MYVAR=$( ls *bed )   # assign the output of a command to a variable
MYVAR=`ls *bed`
```

## Build-in variables

```
$PWD      # current working directory;
$PATH     # default path to executables
```

```
./shellscript.sh arg1 arg2 # Any string after a script is passed to the script as a build-
in variable
```

```
$1=arg1
```

```
$2=arg2
```

# Control structures: if-else

---

## If-else

```
if [condition]
then
    statement1
else
    statement2
fi
```

## example

```
#!/bin/bash

N=$1
M=$2
if [ $N -eq $M ]
then
    echo "Hello world!"
else
    echo "Good bye world!"
fi
```

# The 'test' command = '[' = '['

Statement	Description
test EXPRESSION	Test if an expression is true
[ EXPRESSION ]	Short hand for test, used on all POSIX shells
[[ EXPRESSION ]]	Short hand for test, available with newer shells like bash, ksh, zsh

Expression	Meaning
&&	Logical AND
	Logical OR
-eq	Equality check
-ne	Inequality check
-lt	Less Than
-le	Less Than or Equal
-gt	Greater Than
-ge	Greater Than or Equal

# File checks

---

## Example: Test if a file exists

```
#!/bin/bash

FILE=$1

if [ -e $FILE ]
then
    echo "$FILE exists."
else
    echo "$FILE doesn't exist."
fi
```

# run

\$ Shellscript.sh /etc/config

```
# Check if a file exists
[ -e $FILE ]

# Check if a file is a regular file
[ -f $FILE ]

# Check if a file is a directory
[ -d $FILE ]

# Check if a file is empty
[ -z $FILE ]

# Check if a file is not empty
[ -n $FILE ]
```



# Control structure: for-loop

---

## for loop

```
for iteration
do
    something
done
```

## example

```
#!/bin/bash

for i in {1..10..1}
do
    echo "Hello $i world!"
done
```

# Looping over a list of files

---

## Example: Iteration over a list of files ending with '\*bed'

```
#!/bin/bash

for i in $(ls *bed)
do
    wc -l $i
done
```

\* \$( ) command substitution/ content is evaluated

# Control structure: while-loop

## While loop

```
while [condition]
do
    something
done
```

## example

```
#!/bin/bash

COUNTER=0
while [ $COUNTER -le 5 ]
do
    echo "Welcome $COUNTER times"
    COUNTER=$(( $COUNTER + 1 ))
done
```

\* `$(( ))` content is evaluated in numeric context

# Operating on a file line-by-line

**Example: Subtract and add 200 to a set of fields (e.g. genomic coordinates)**

```
#!/bin/bash

FILE=$1

Cut -f 1-3 $FILE | while read CHR START END
do
    echo $CHR $(( $START - 200 )) $(( $END + 200 ))
done
```

## More on shell scripting

---

<https://ryanstutorials.net/bash-scripting-tutorial/>

Thank you!

