

Introduction to UNIX/Linux commands

ISI network

27-28 November
Montpellier

ISI network (IRD Scientific Computing)

Communities of Practice (CoP) network open in February 2024



104 members

*volunteering
collegiality
sharing information
respect for others*

IRD staff or partners who produce, manage or use scientific digital tools

Exchange



skills development

Share



*issues
needs*

Mutualize



*training
documentation*

2 first CoPs :

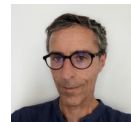
- Collaborative software development
- Health research data management

Organisation:

- Shared training WG
- Coordination: Coordination Committee (network coordinator, coordinators for each of the CoPs), videoconferences, annual seminar, etc.

Collaborative space: discussion forum, wikis

hugo.catherine@ird.fr





Join us! <https://copnumeriquescientifique.ird.fr/>

Training material

Session 2024



Python : <https://e-formation.ird.fr/course/view.php?id=264>

- 18th to 20th of november



GIT : <https://e-formation.ird.fr/course/view.php?id=265>

- 25th of november



UNIX/Linux : <https://e-formation.ird.fr/course/view.php?id=267>

- 27th and 28th of november

with the support of the IRD's SDTQVT department ISI



The educational material used for these teachings is made available under the terms of the Creative Commons Attribution - NonCommercial - ShareAlike (BY-NC-SA) 4.0 International license."
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

UNIX/Linux Trainers



Alexis Dereeper
Bioinformatics



Ndomassi TANDO
System Engineer



Jacques Dainat
Bioinformatics



Laurent Demagistri
Scientific computation
and geomatic



Vincent Manzanilla
Bioinformatics



UNIX/Linux course materials



Content licensed under CC BY-NC-SA 4.0 & created in 2011 by :

- Christine Tranchant
- Bruno Granouillac



updated by

- Gautier Sarah
- Christine Tranchant
- Ndomassi Tando
- Alexis Dereeper
- Jacques Dainat



2018



2024



2024



2024



2024



The educational material used for these teachings is made available under the terms of the Creative Commons Attribution - NonCommercial - ShareAlike (BY-NC-SA) 4.0 International license."
<http://creativecommons.org/licenses/by-nc-sa/4.0/>



Learning objectives

The objective

Run your own analysis using Linux !

After this course, you will be able to :

- Know the main Linux commands
- Move into the Linux file tree : *pwd, ls, cd, mkdir* etc.
- Connect to a Linux server and transfer data : *ssh, wget*
- Work with text files: *head, tail, sort, cut, wc, grep...*
- Chain and combine commands
- Run programs from the command line
- Create a basic script

Introduction

What is Linux?

- **Operating system** known for :
 - its security and stability
 - its frequent updates
 - its (no) fees and (mostly)openSource softwares
- Created in 1991 by ***Linus Torvalds***
- Based on Unix (1969)
- Linux source code ***opensource*** and ***free*** : copy, modify, redistribute



What is Linux?

- **Robust et multi-plateform OS**
(computer, server, android....)

- **Multi-users system**

Several users can work simultaneously

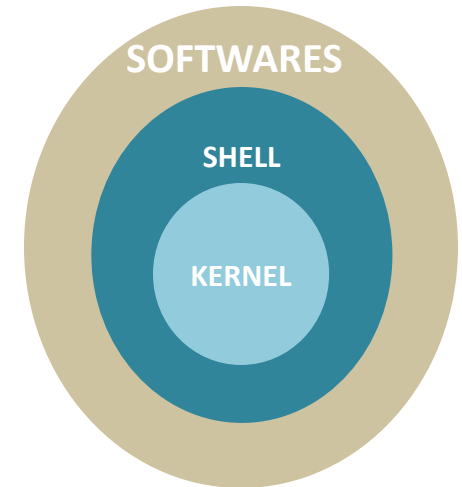
- **Multi-tasking system (processes/programs)**

Every user can run several programs at the same time



Linux distribution

Distribution : Kernel + Shell + Softwares

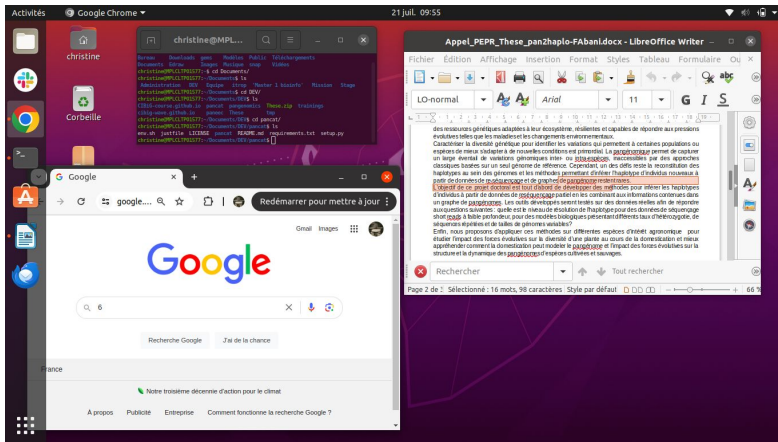


How to use Linux?

2 ways



Graphical User Interface
personal computer

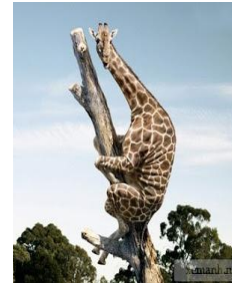


How to use Linux?

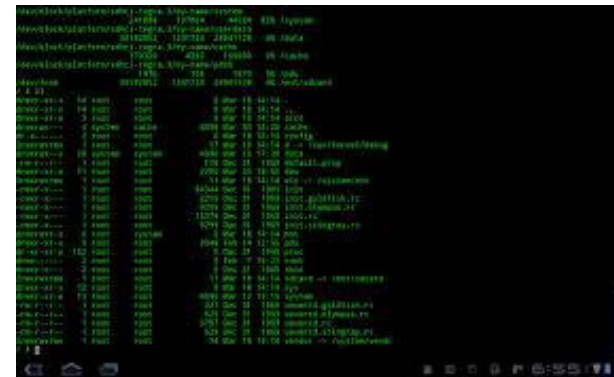
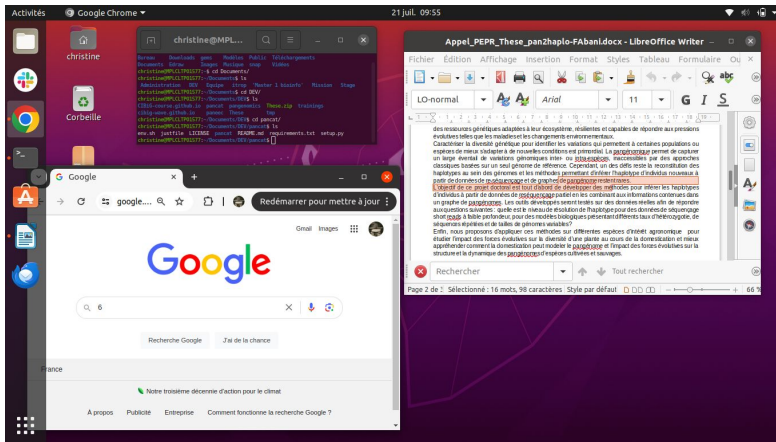
2 ways



Graphical User Interface
personal computer



Command-Line Interface
through a terminal
personal computer, server



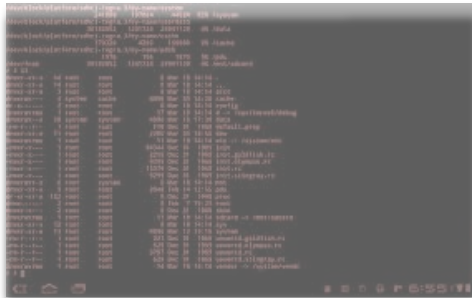
No graphical interface

Why using Linux?

- Numerous fast and powerful programs
- Easy to link commands and programs (workflow)
- Numerous scientific softwares available
- 90% of servers under Linux

Why using Linux?

- Numerous fast and powerful programs
- Easy to link commands and programs (workflow)
- Numerous scientific softwares available
- 90% of servers under Linux



No graphical interface

Command line

Why using Linux?



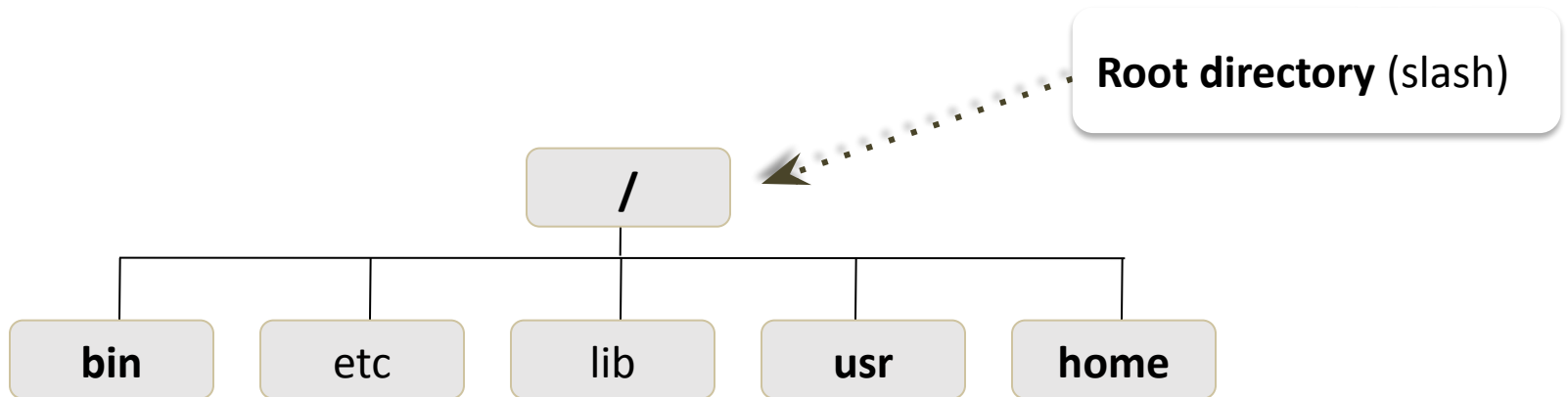
Need to practice

⇔ **Need important investments to get good results quickly**

Linux File tree

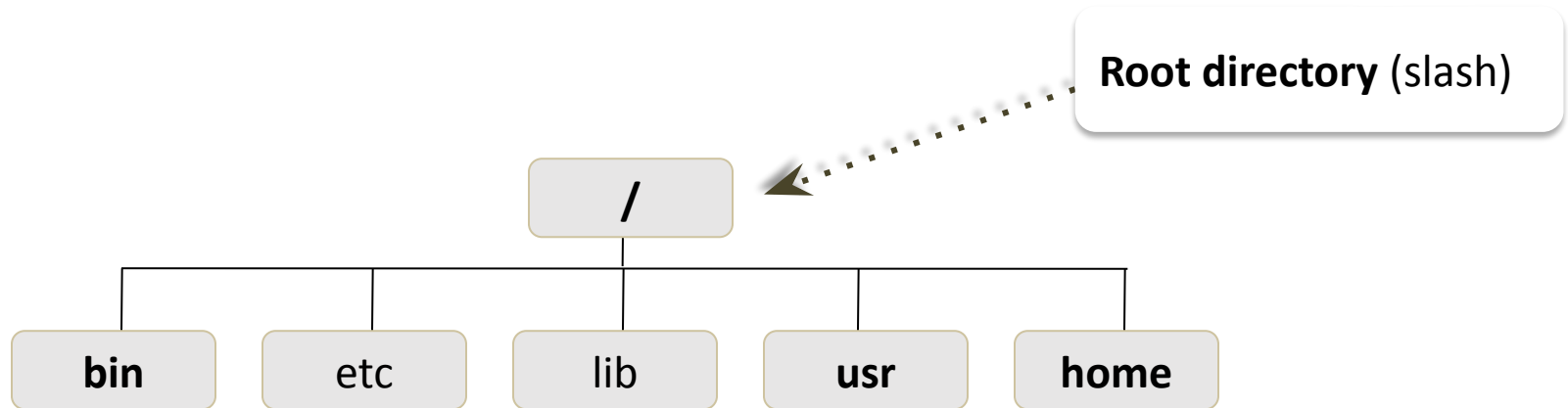
The file tree - Filesystem

- Directory structure starts at the root directory called “/” (slash)



The file tree - Filesystem

Main directories

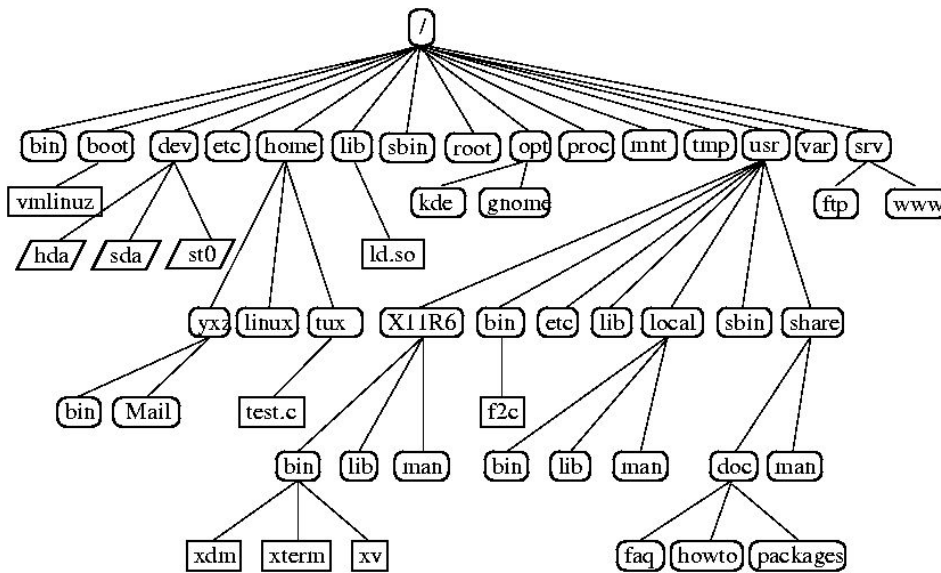


/bin	Main commands, shell, programs
/usr, /opt	Applications and user libraries
/usr/bin	Other commands
/home	User directory (one per user, name= login)

File Path

Path

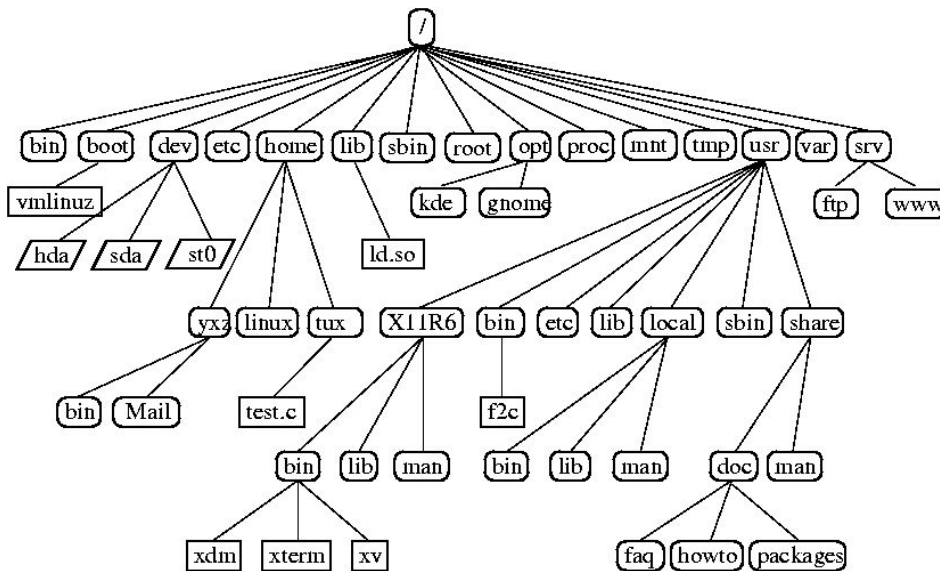
location of a file/directory in the LINUX file system



File Path

Path

location of a file/directory in the LINUX file system



- Absolute Path



- Relative Path



File Path: Absolute path

Path

location of a file/directory in the LINUX file system



Absolute Path

- ✓ Complete path of a file starting from the root directory /



Relative Path

File Path: Absolute path

Path

location of a file/directory in the LINUX file system



Absolute Path



Relative Path

✓ Complete path of a file starting from the root directory /

➡ ***Always starts with /***

➡ **Always right wherever the user is**

File Path: Relative path

Path

location of a file/directory in the LINUX file system



Absolute Path

✓ Complete path of a file starting from the root directory /

➡ *Always starts with /*

➡ Always right wherever the user is



Relative Path

✓ Path related to the present working directory - *where the user is working*

File Path: Relative path

Path

location of a file/directory in the LINUX file system



Absolute Path

✓ Complete path of a file starting from the root directory /

➡ *Always starts with /*

➡ Always right wherever the user is



Relative Path

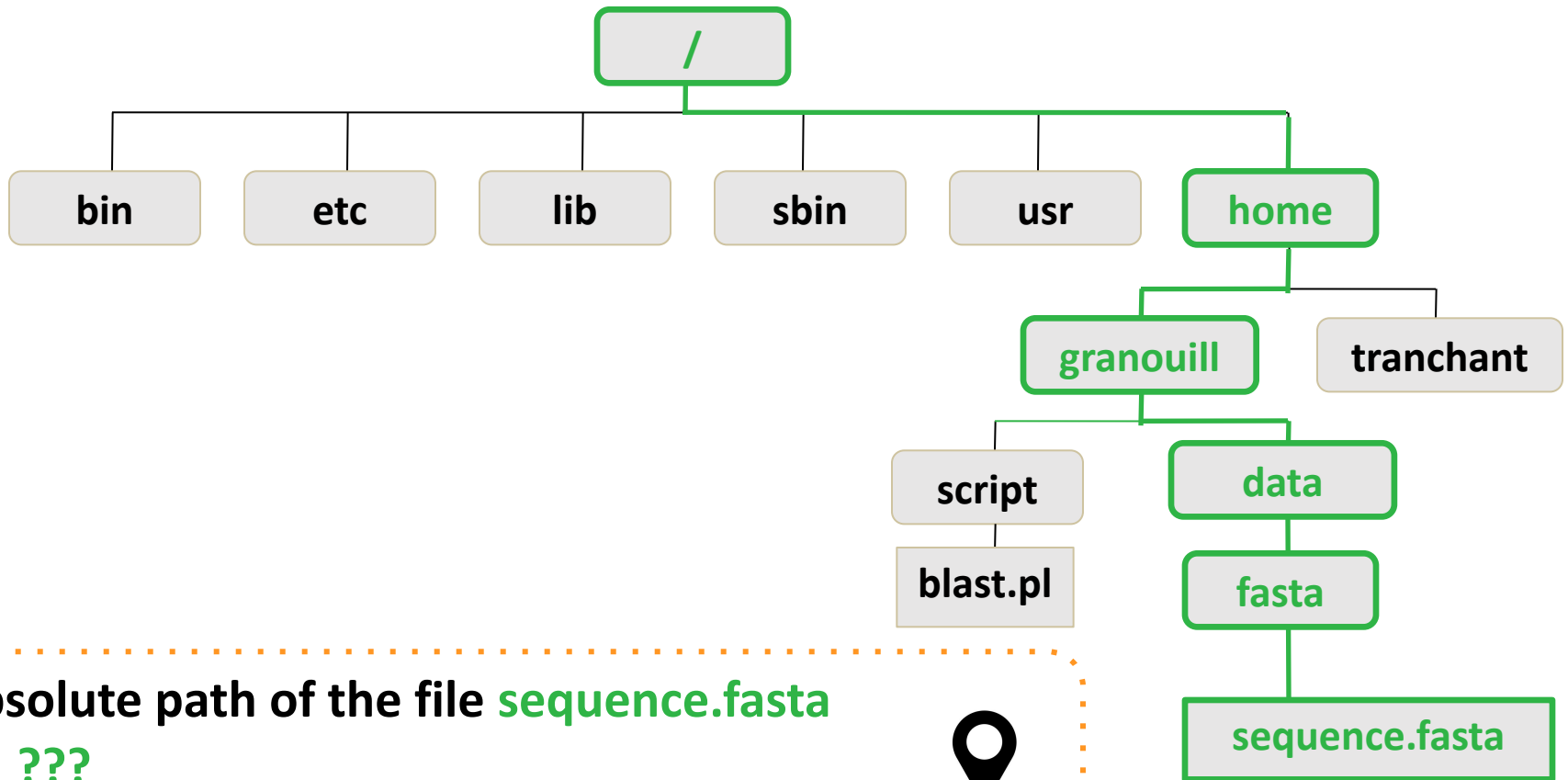
✓ Path related to the present working directory - *where the user is working*

➡ ***Never starts with /***

➡ ***Depends on where the user is working***

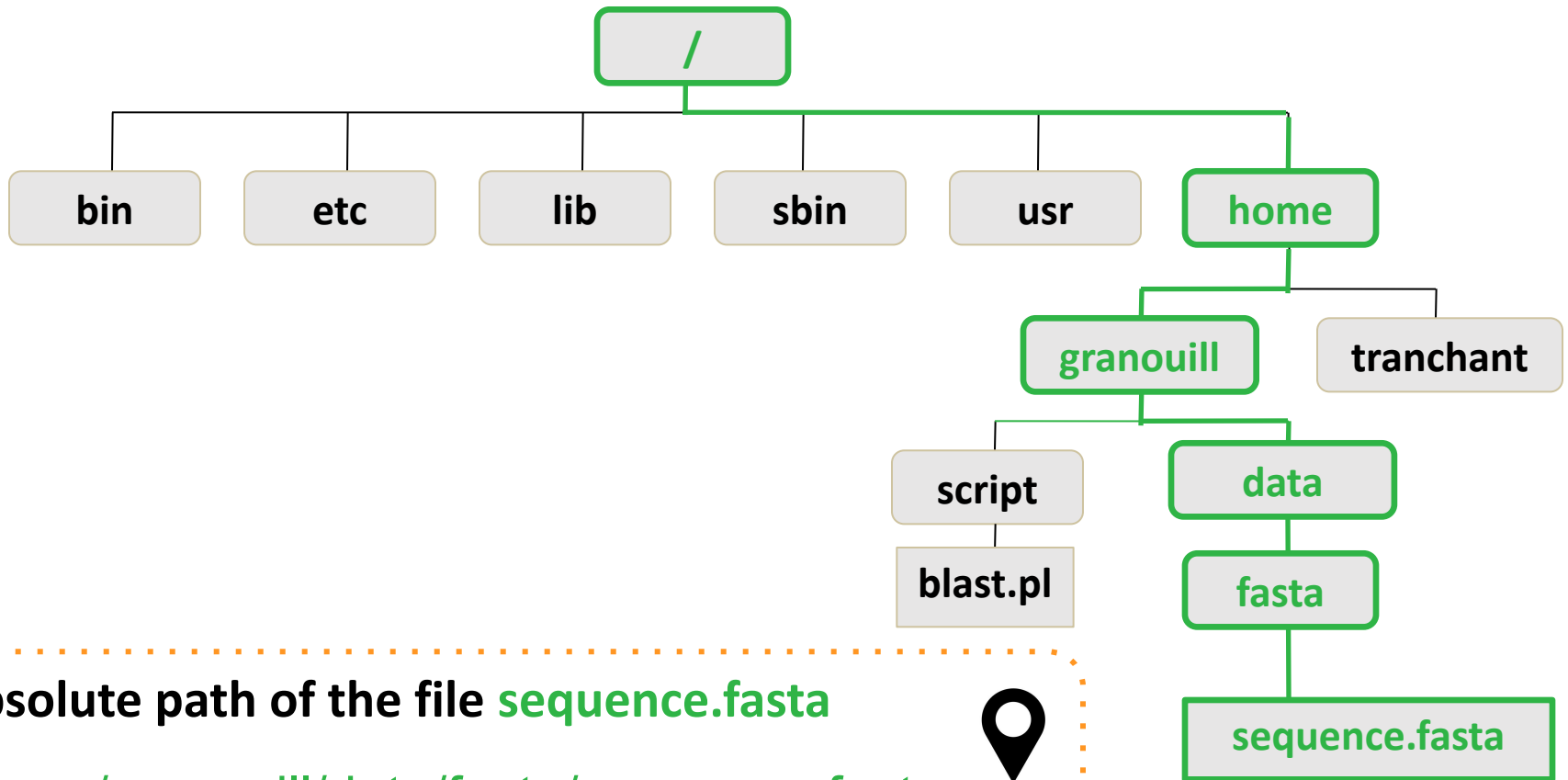
Example 1 of absolute Path

- Always starts with / (root directory)
- Always works wherever user is working



Example 1 of absolute Path

- Always starts with / (root directory)
- Always works wherever user is working



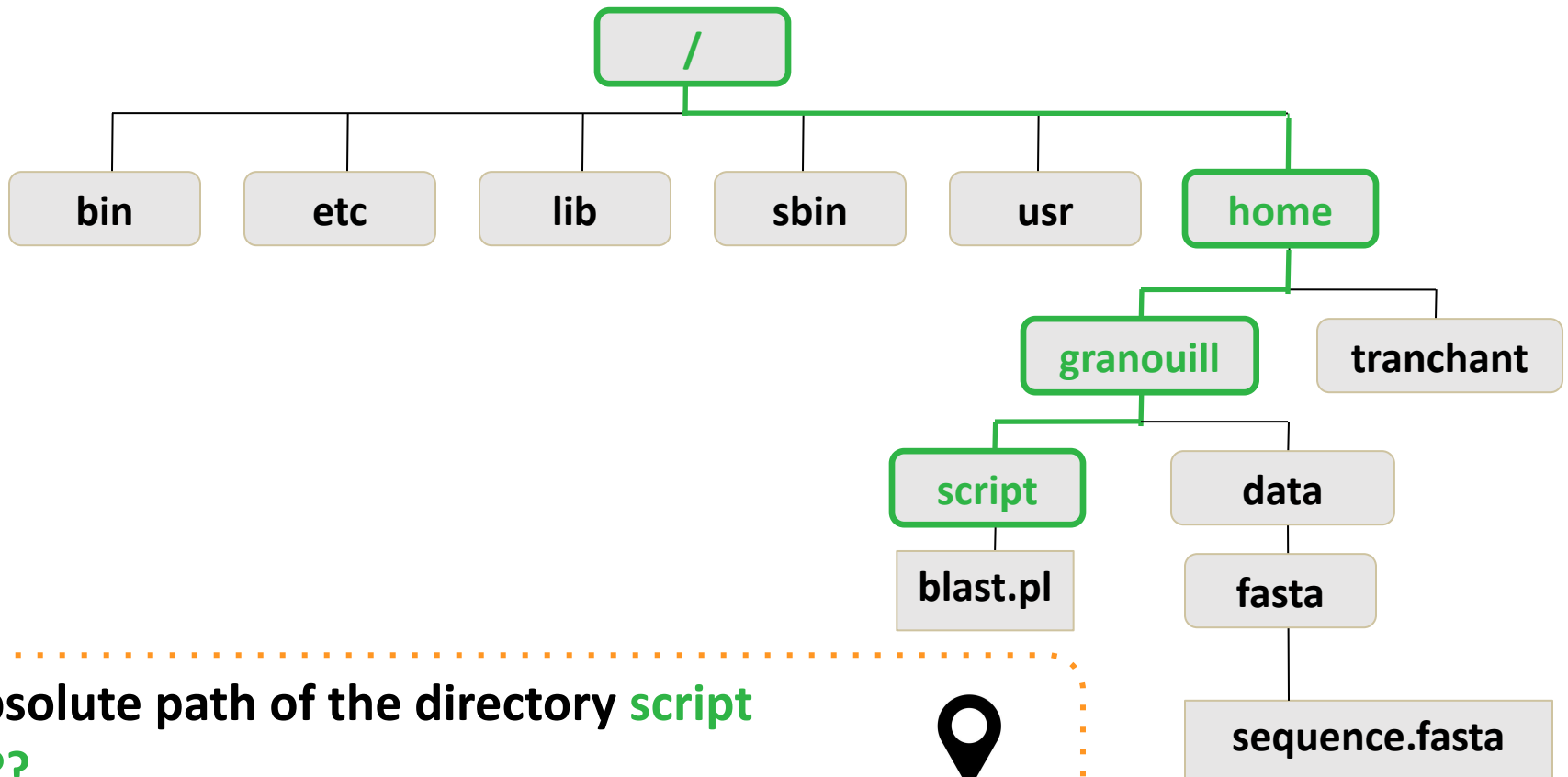
Absolute path of the file **sequence.fasta**

/home/granouill/data/fasta/sequence.fasta



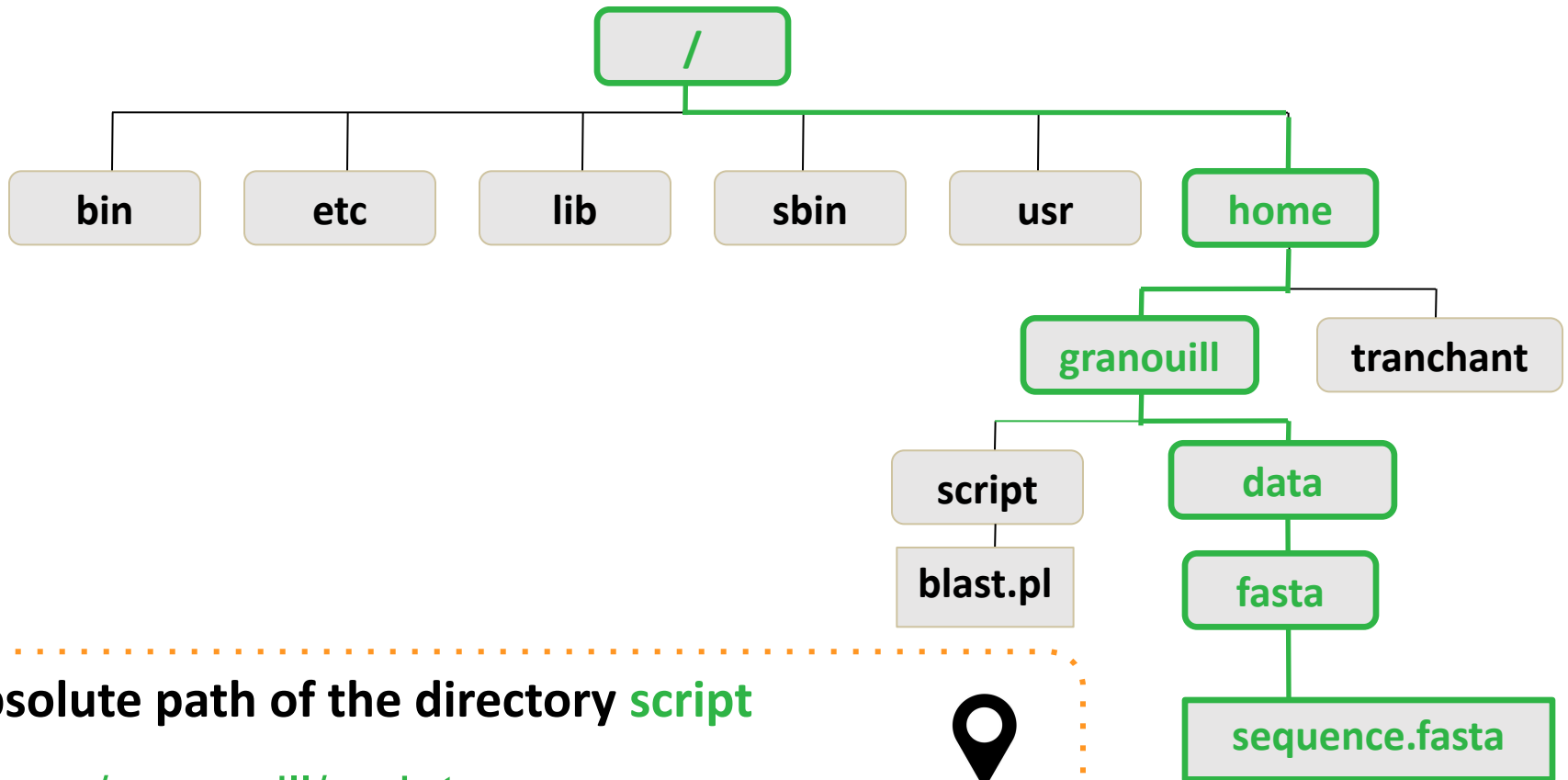
Example 2 of absolute Path

- Always starts with / (root directory)
- Always works wherever user is working



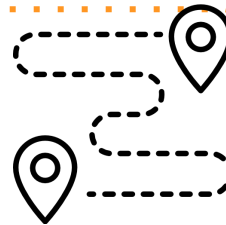
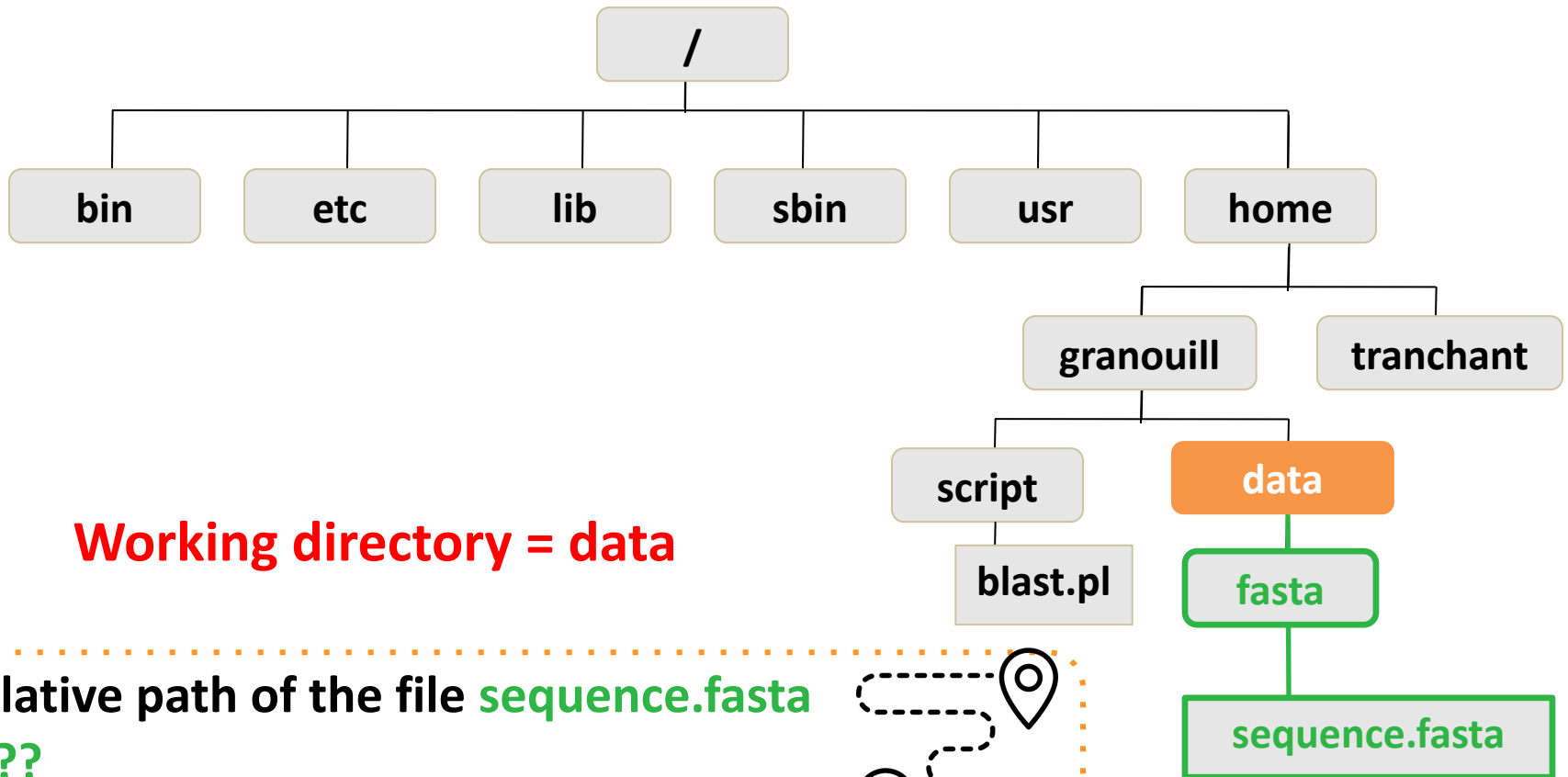
Example 2 of absolute Path

- Always starts with `/` (root directory)
- Always works wherever user is working



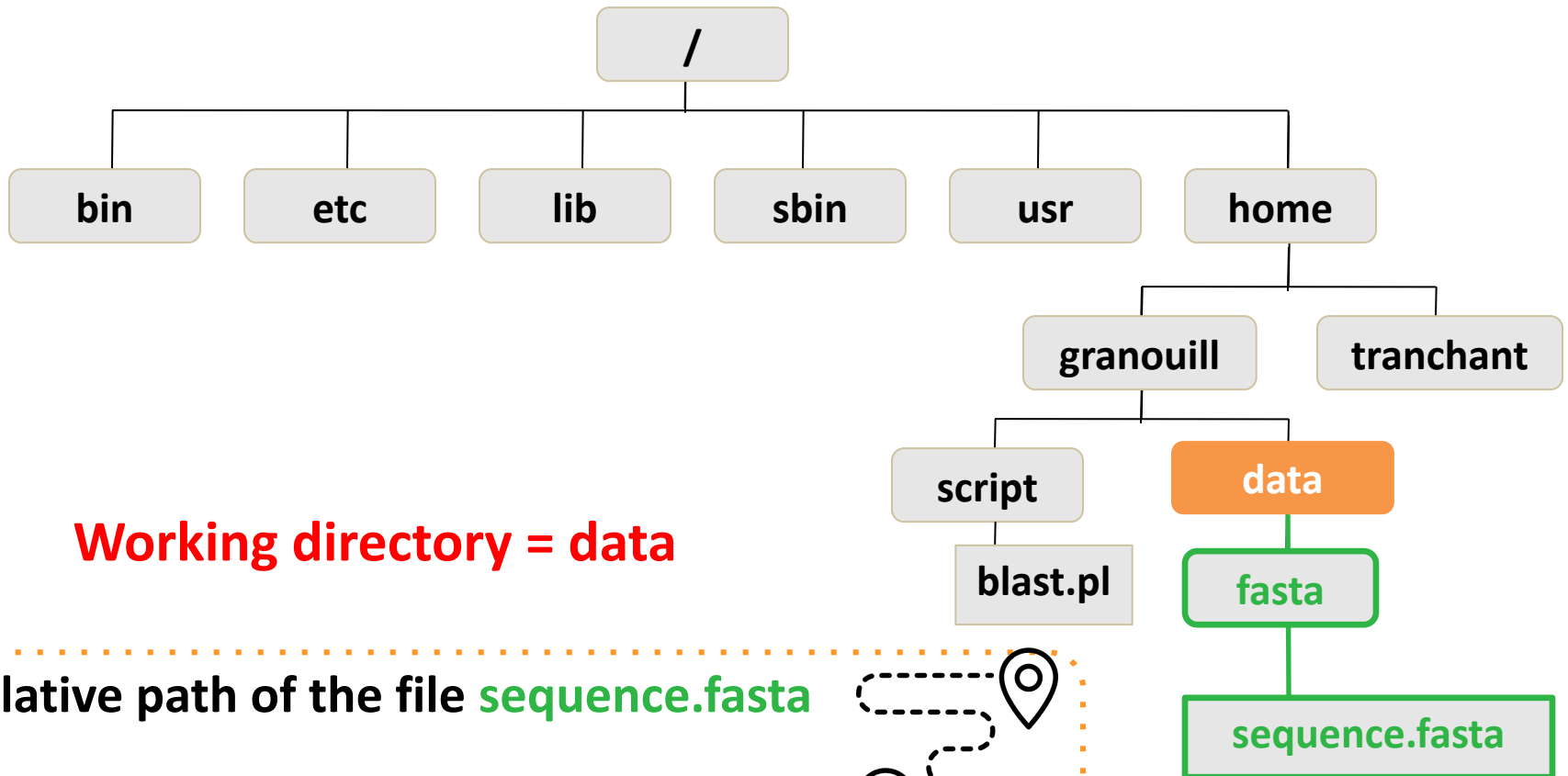
Example 1 of relative Path

- Path related to the present working directory
- **Never starts with /**



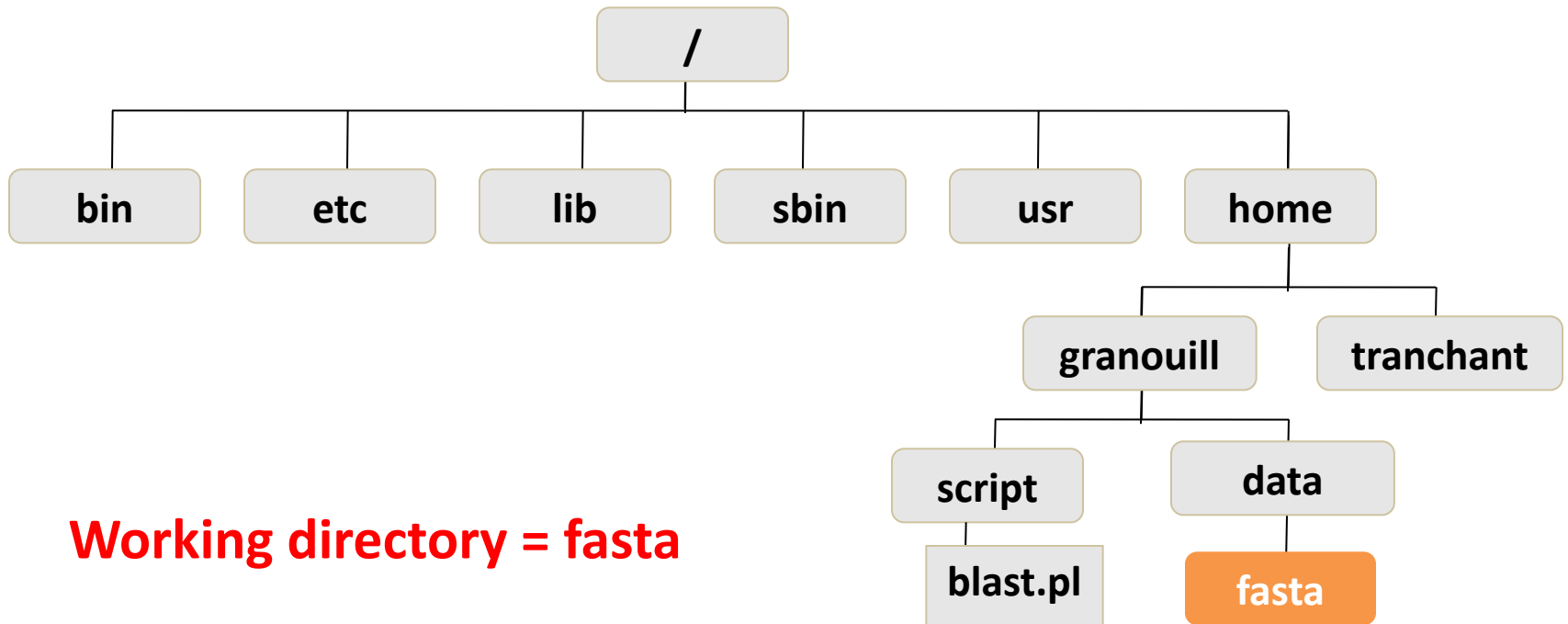
Example 1 of relative Path

- Path related to the present working directory
- **Never starts with /**



Example 2 of relative Path

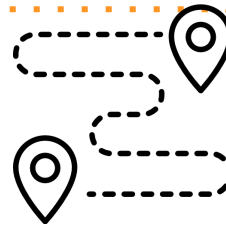
- Path related to the present working directory
- **Never starts with /**



Working directory = fasta

Relative path of the file **sequence.fasta**

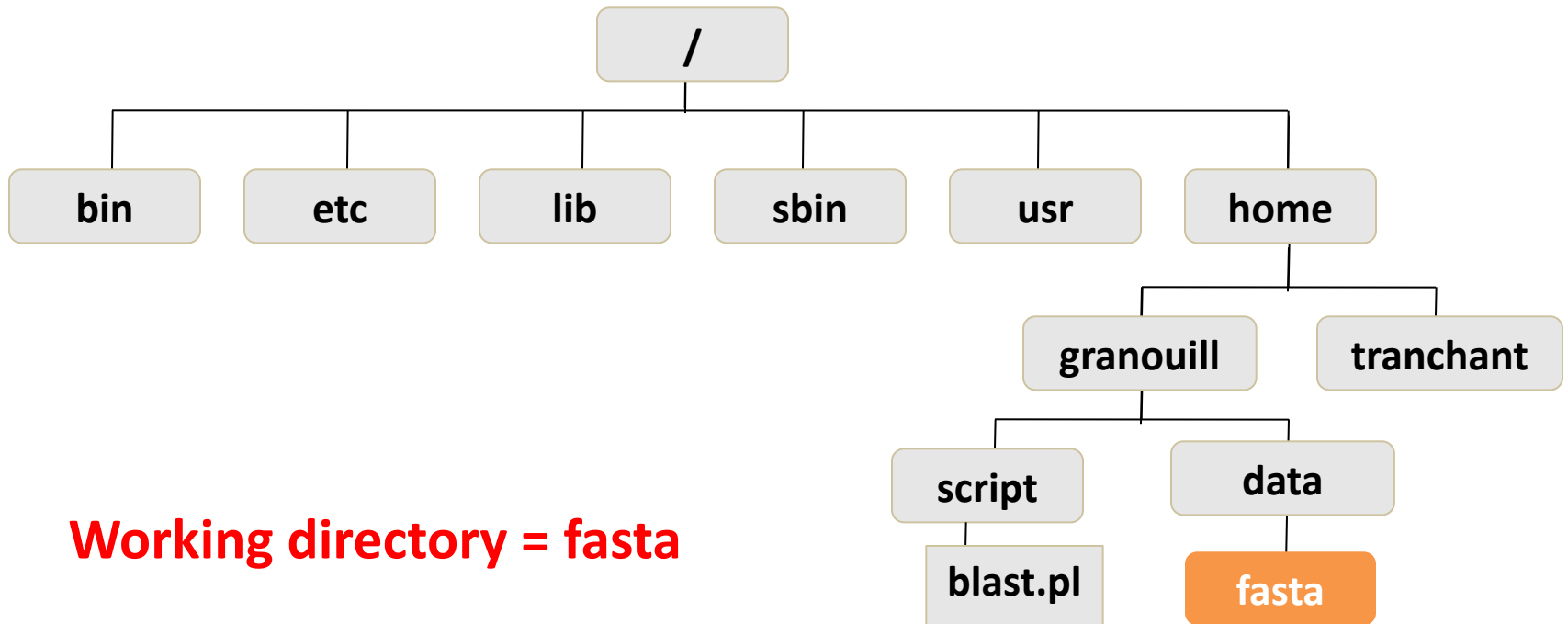
???



sequence.fasta

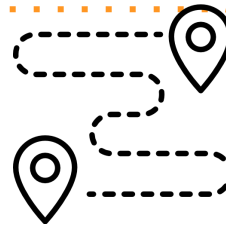
Example 2 of relative Path

- Path related to the present working directory
- **Never starts with /**



Working directory = fasta

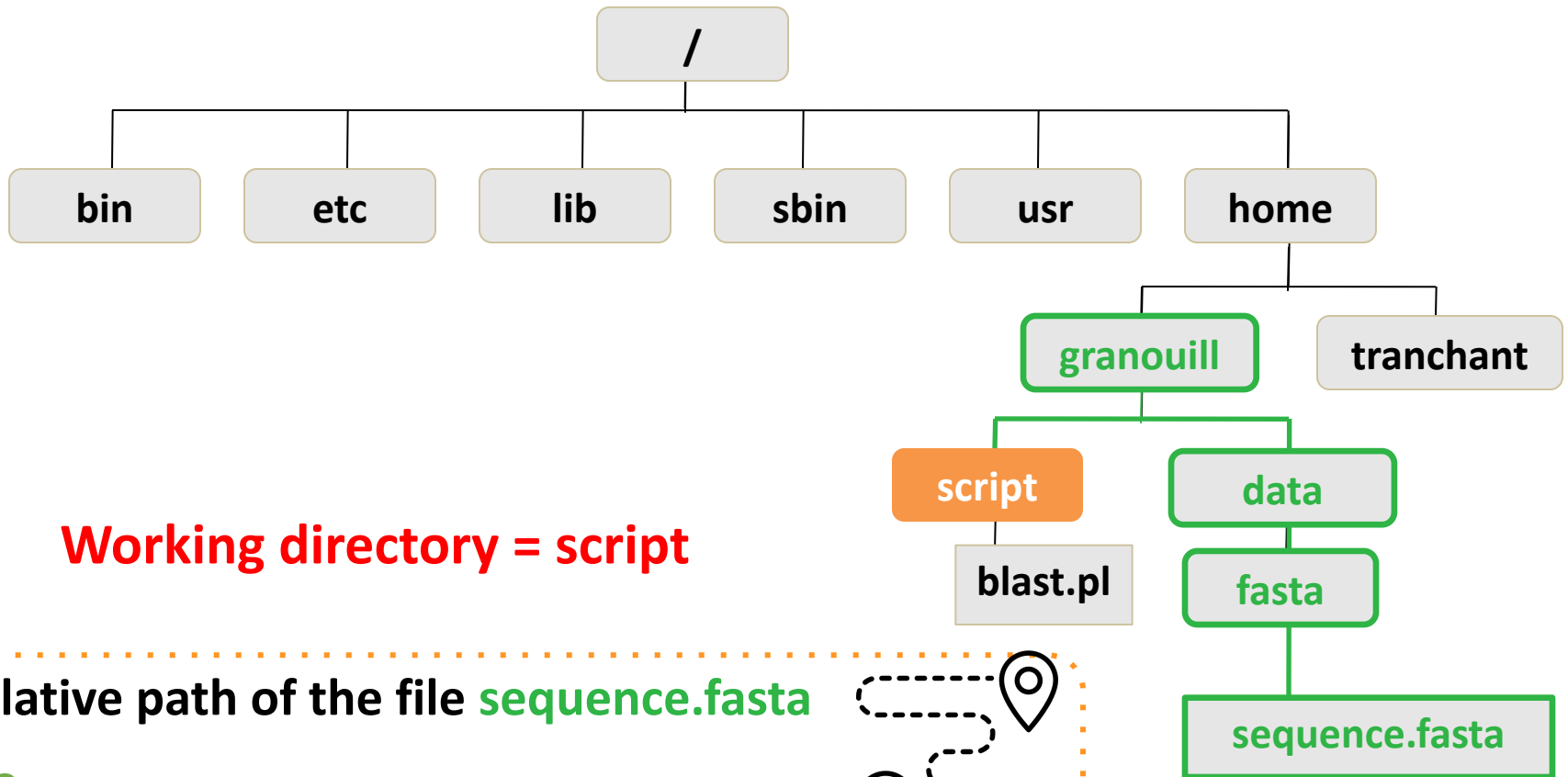
Relative path of the file **sequence.fasta**
sequence.fasta



sequence.fasta

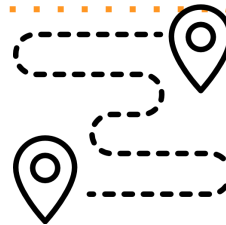
Example 3 of relative Path

- Path related to the present working directory
- **Never starts with /**



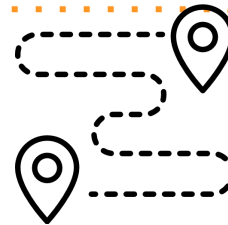
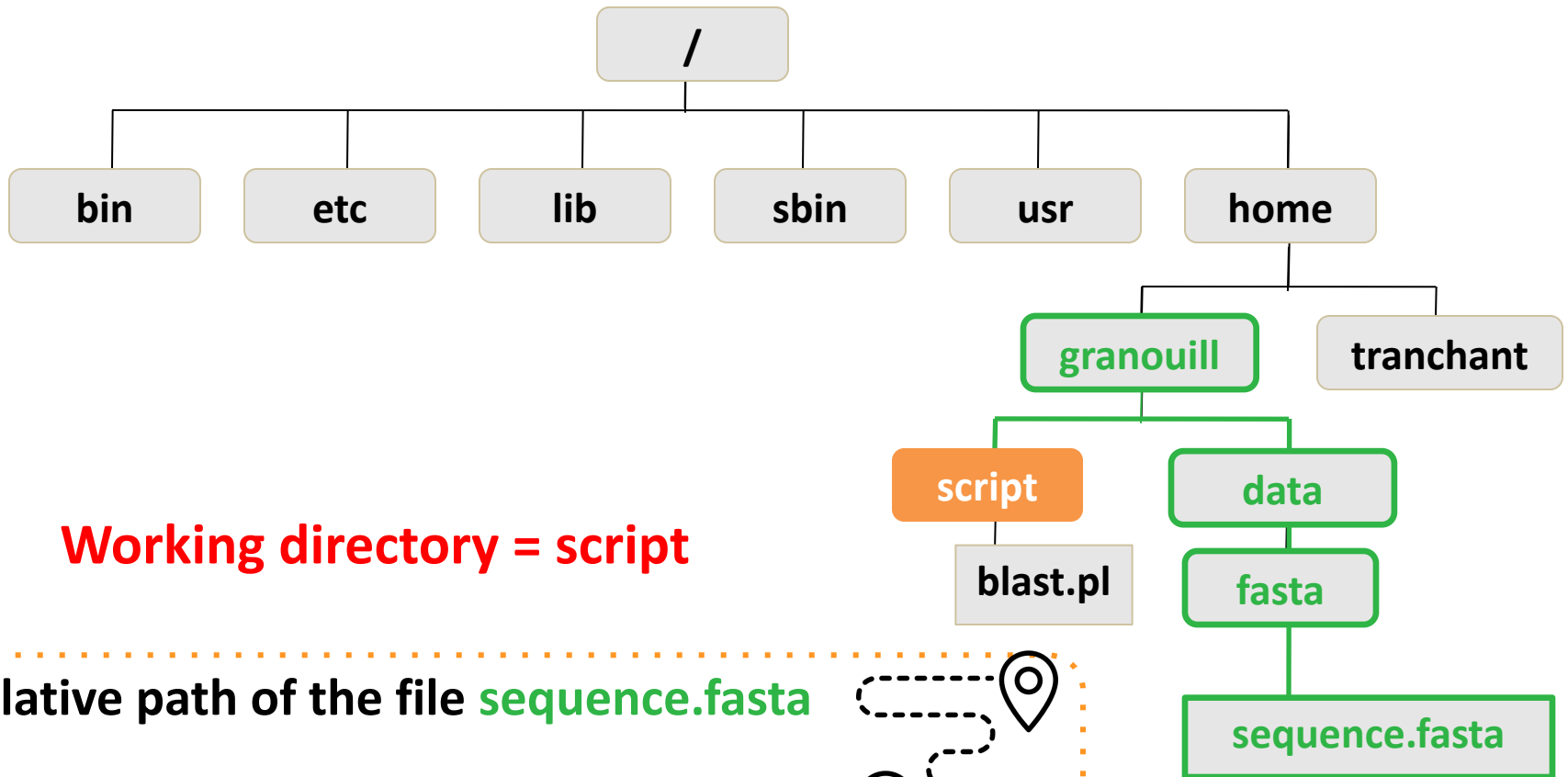
Relative path of the file **sequence.fasta**

???



Example 3 of relative Path

- Path related to the present working directory
- **Never starts with /**



Prompt and first commands

`pwd, ls commands`

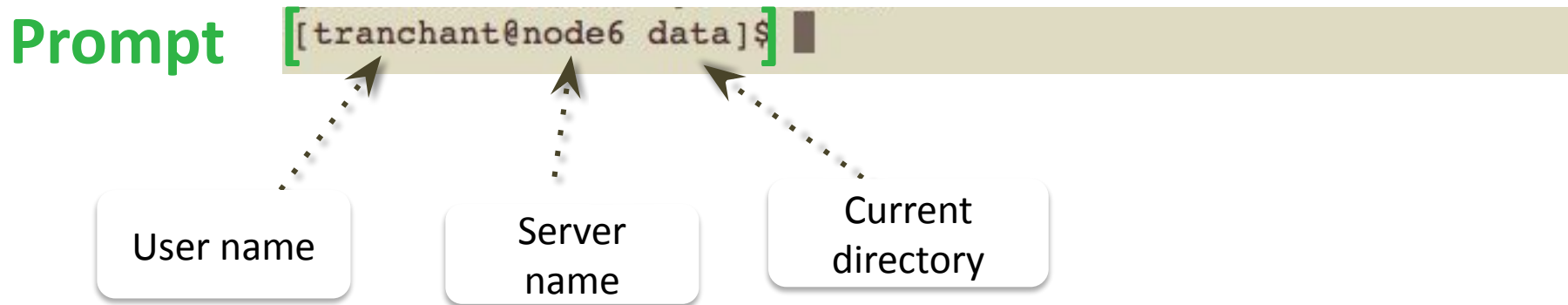
The prompt

Always on the terminal, just before where user types commands

Prompt `[tranchant@node6 data]$` ■

The prompt

Always on the terminal, just before where user types commands



Command syntax

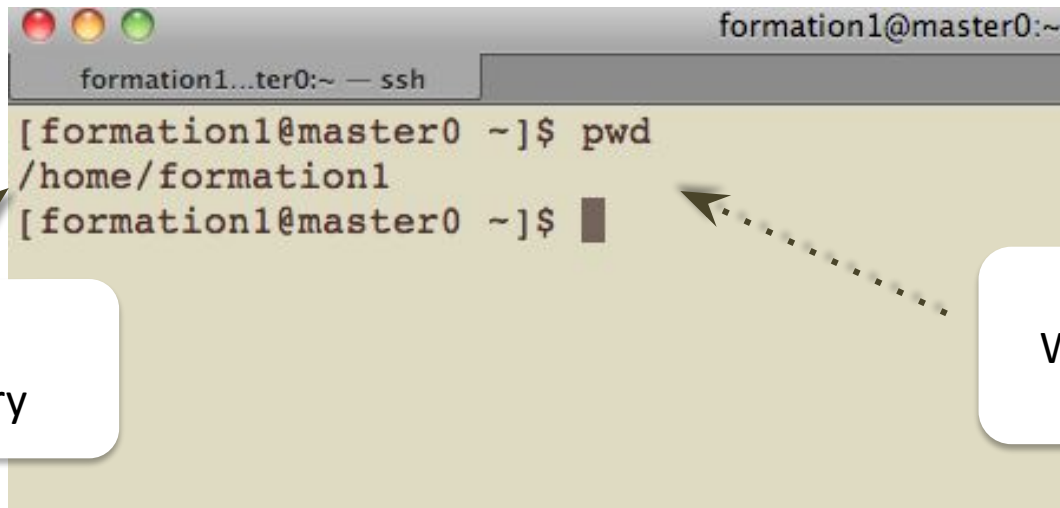
command [-options] [arguments]

Your first command “*pwd*”

pwd

Present Work Directory

*Print the name of the current directory
(the full path)*

A terminal window titled 'formation1@master0:~' with a tab labeled 'formation1...ter0:~ — ssh'. The prompt is '[formation1@master0 ~]\$'. The command 'pwd' has been entered, and the output is '/home/formation1'. The prompt is now '[formation1@master0 ~]\$' followed by a cursor. Dotted arrows point from the output and the command to callout boxes.

```
formation1@master0:~  
[formation1@master0 ~]$ pwd  
/home/formation1  
[formation1@master0 ~]$
```

Name of the
current directory

Command
Without option and
argument

Second command “ls”

ls
list

List the content of the current directory



A terminal window titled "formation1@master0:~ — ssh — 97x37" with a sub-tab "formation1...ter0:~ — ssh". The prompt is "[formation1@master0 ~]\$". The command "ls" has been entered, and the output shows "data" in blue and "scripts" in purple. A dashed arrow points from the "ls" command in the prompt to the text "Command without option and argument". Another dashed arrow points from the "data" output to the text "List all the files in the current directory (by default)".

```
[formation1@master0 ~]$ ls  
data  scripts
```

List all the files in the current directory (by default)

Command without option and argument



Practice

set up environnement,
prompt, pwd

1

Go to [the set up environment practice](#) on our
website

Second command + option " ls -l"

ls -l
list long

list files with more information about each file

Command with the option **-l** and a **directory name** given as argument

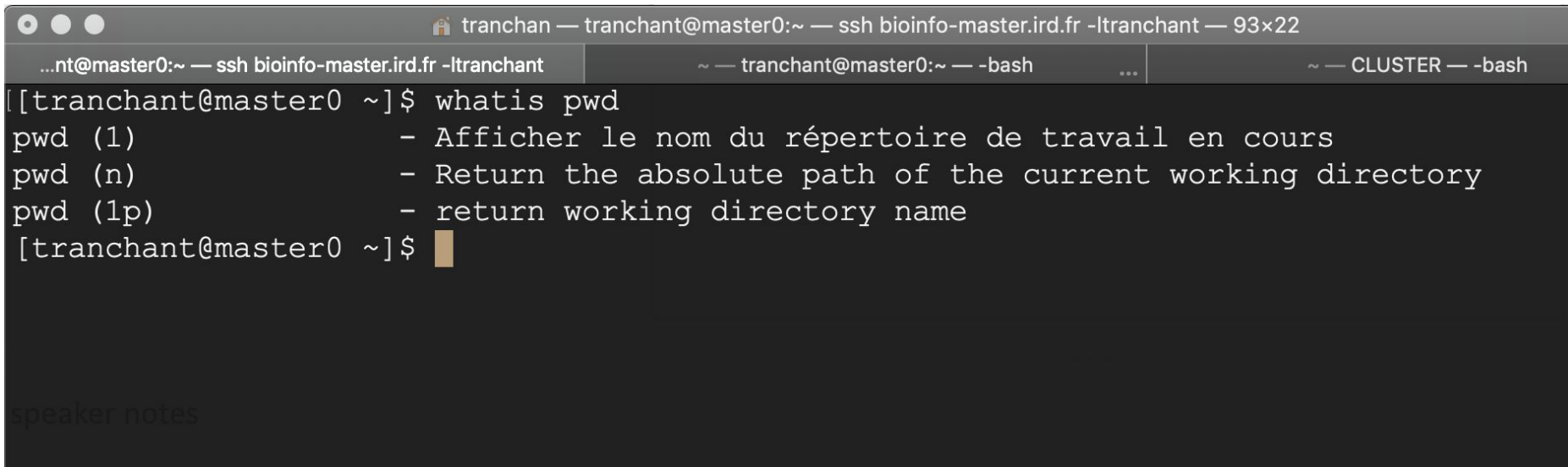
```
[formation1@master0 ~]$ ls -l /home/
total 312
drwx-----  6 abate      sat      4096 12 mars   2012 abate
drwx-----  5 adam       ggr      4096 23 mars   2012 adam
drwx----- 31 admin     admin    4096  3 août   11:35 admin
drwx-----  9 alizon    ete      4096 21 août   14:23 alizon
drwx----- 12 alvaro-wis effecteurs 4096 17 juin   16:19 alvaro-wis
drwx-----  4 auguy     rhizogenesis 4096  2 mars   2012 auguy
drwx-----  5 ayoubas  team1    4096 13 avril  2012 ayoubas
drwx-----  5 beule     bdp      4096  8 oct.   17:49 beule
drwx-----  9 bouniol   ggr      4096  2 oct.   15:00 bouniol
drwx----- 10 castillo  bdp      4096 10 oct.   15:55 castillo
```

Display the long format listing of all files in the directory

Few basic commands

How to get help about one command

- with the 'option *--help* ou *-h* *ls --help* *blastn -h*
- with the command *man* *man ls*
- with the command *what is* *what is ls*



```
tranchan — tranchant@master0:~ — ssh bioinfo-master.ird.fr -ltranchant — 93x22
...nt@master0:~ — ssh bioinfo-master.ird.fr -ltranchant  ~ — tranchant@master0:~ — -bash  ...  ~ — CLUSTER — -bash
[[tranchant@master0 ~]$ what is pwd
pwd (1)          - Afficher le nom du répertoire de travail en cours
pwd (n)          - Return the absolute path of the current working directory
pwd (lp)         - return working directory name
[tranchant@master0 ~]$
```

speaker notes

Few basic commands

Basics

pwd

Display the full path of the current directory

ls

List all files/directories

ls -l

Display all files (Long listing)

Commands to navigate into the file system

cd command

The “cd” command

cd

Change Directory

Move from the current directory into a new directory

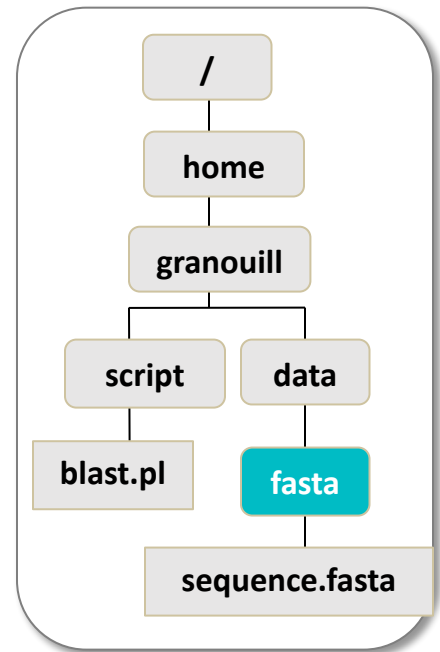
```
cd DIRECTORY_NAME  
    absolute ou relative path
```

The “cd” command: example 1

absolute path

cd DIRECTORY_NAME - absolute path

Absolute path of the directory **fasta**



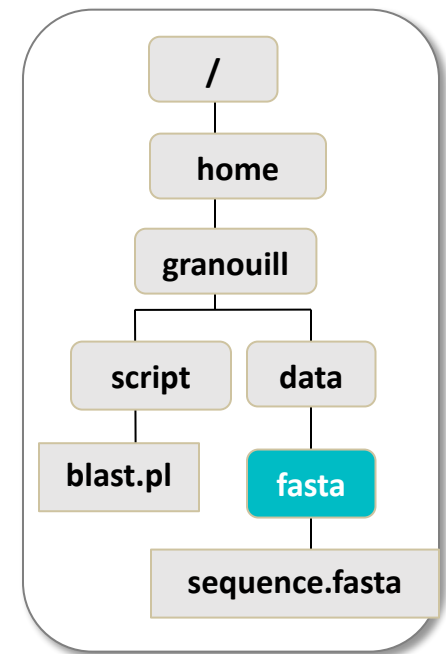
The “cd” command: example 1

absolute path

cd DIRECTORY_NAME - absolute path

Absolute path of the directory **fasta** 

cd /home/granouill/data/fasta

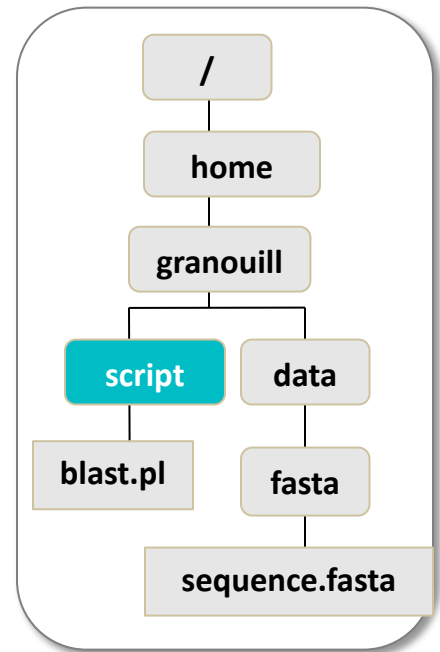


The “cd” command: example 2

absolute path

cd DIRECTORY_NAME - absolute path

Absolute path of the directory **script**



The “cd” command: example 2

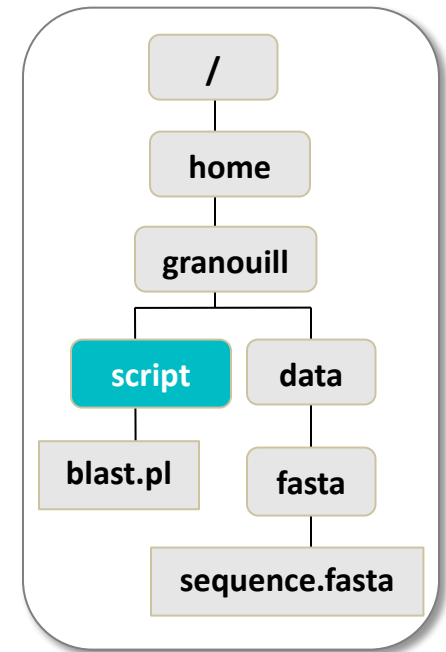
absolute path

cd DIRECTORY_NAME - absolute path

Absolute path of the directory `script`



cd /home/granouill/script

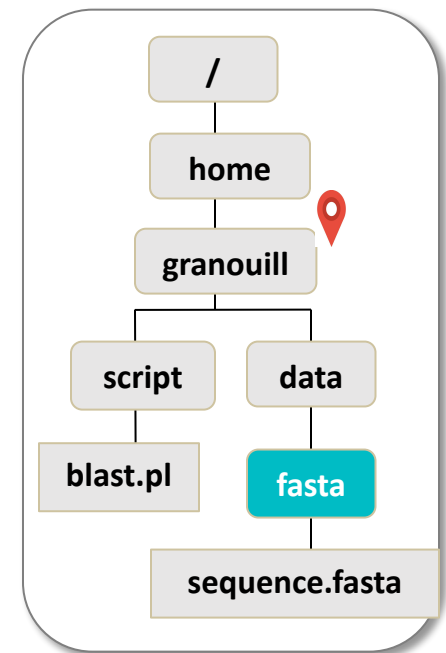


The “cd” command: example 1

relative path

cd DIRECTORY_NAME - relative path

Relative path of the directory **fasta**



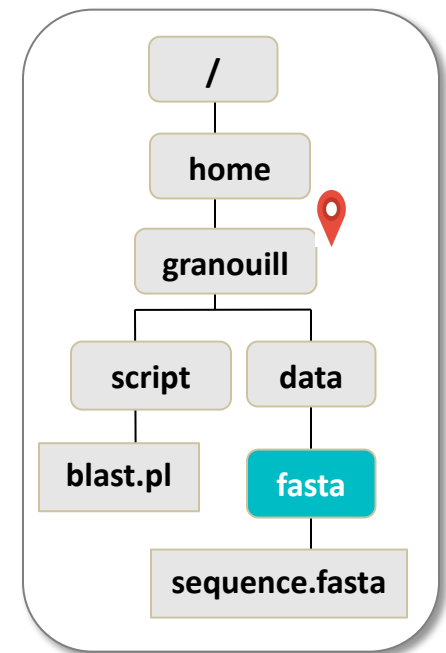
The “cd” command: example 1

relative path

cd DIRECTORY_NAME - relative path

Relative path of the directory **fasta**

cd data/fasta



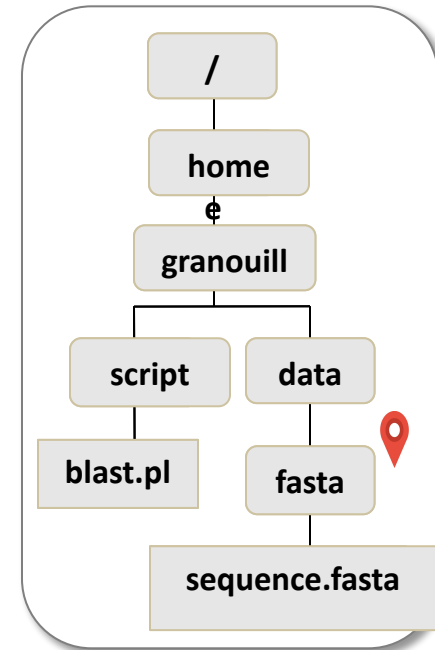
The “cd” command: shortcuts relative path

cd DIRECTORY_NAME - relative path



Command	Go to
<code>cd</code>	<i>home directory</i>
<code>cd ..</code>	Parent directory
<code>cd ../..</code>	?
<code>cd -</code>	?

Go to home
directory
One folder up



The “cd” command: shortcuts relative path

cd DIRECTORY_NAME - relative path

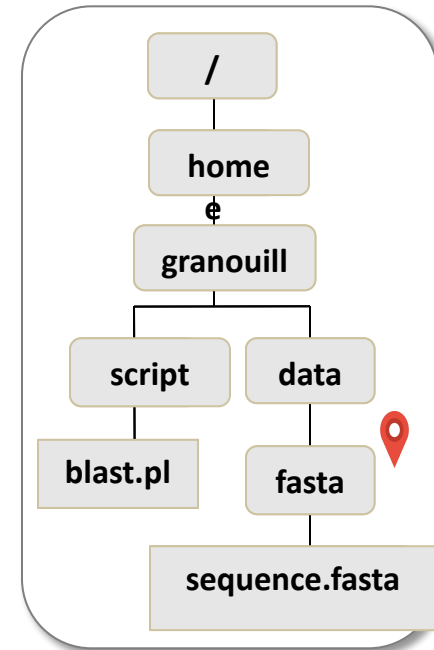


Command	Go to
<i>cd</i>	<i>home directory</i>
<i>cd ..</i>	Parent directory
<i>cd ../..</i>	?
<i>cd -</i>	?

Go to home
directory

One folder up

2 folders up



The “cd” command: shortcuts relative path

`cd DIRECTORY_NAME` - *relative path*



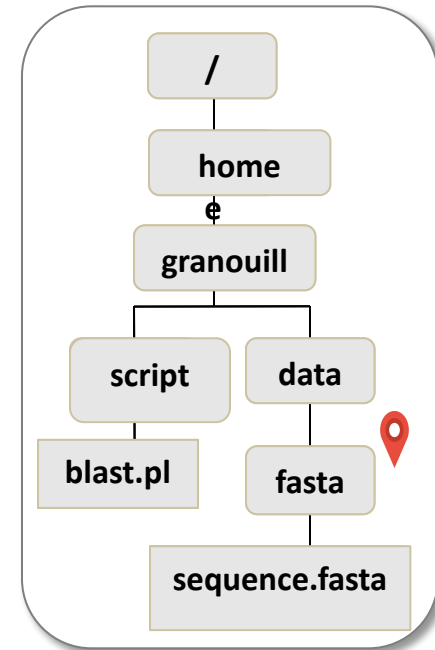
Command	Go to
<code>cd</code>	<i>home directory</i>
<code>cd ..</code>	Parent directory
<code>cd ../..</code>	?
<code>cd -</code>	?

Go to home
directory

One folder up

2 folders up

Go back to previous directory



Rules for files and directories naming

- ✓ Linux is case sensitive

Sequence.fasta **≠** **SEQUENCE.fasta** **≠** **sequence.fasta**

- ✓ Only ROMAN letters, numbers and _ -

- ✓ No space, accent or special symbol

& ~ # " ' { ([| ` \ ^ @)] } \$ * % ! / ; , ?

- ✓ No need to use filename extension (.txt), just to improve readability of filenames.

TIPS - Useful terminal shortcuts

- <Ctrl> + C** Stop the current process/command
- <Tab>** Auto-complete commands, files or directories you are typing
- <Tab> twice** List all possible completions
- Up arrow** Show the previous command. Press it multiple times to walk back through the history
- Down arrow** Show the next command.
- <Ctrl> + R** Search command history (backward search) matching the characters you are typing.



Practice

ls,cd

2

Go to [the navigating practice](#) on our website

Basics commands to know

pwd

Print the full path of the current directory

ls

Display the list of files in a directory

cd DIR_NAME

Change the working directory

Basics commands to know

pwd	Print the full path of the current directory
ls	Display the list of files in a directory
cd DIR_NAME	Change the working directory

mkdir rep_name	Create a new directory
rm nom_fichier	Remove a directory
cp file1 file2	Make a copy of file1 and call it file2
cp FILE_NAME DIR_NAME	Copy the file FILE_NAME in the directory DIR_NAME, keeping the same name
cp FILE DIR/NEW_FILE	Mix of 2

symbolic links

Allow to attribute another path to a file by pointing to a file name.

It is a shortcut **ln**

```
ln -s the right the wrong
```

Example: `ln -s /opt/jdk-7.01 /opt/jdk`

Save disk space on a system: only the "real" file weights

Commands to display texts

“touch” command

touch <file_name> *Create the file called file_name*

```
1: Terminal ▾  
[dom@dom-XPS-13-9370 ~]$ touch example.txt  
[dom@dom-XPS-13-9370 ~]$
```

“cat” command

cat <file_name>

Displays the content of a file on the screen

Cat

Don't use it with big files!!

```
MacBook-Pro-de-Christine:Data tranchan$ cat Data/Fasta/sequence.fasta
>Gxbjbsjxbjs
CCACCCCTCTTACAGTCTTCACCAAATGTCCTTTAAAACTCCACCTAAAGTATCCAAAGA
CTCGAGAAATGCTGTGCCACAACCAGCTTTTGAGTCATCCATGACCGTTGATCTTCCTTT
GCCCCCAGAGTGGGGCCTAGCACCATCTAGCTACTACTTGCCCTTTCATACCCATCATTGG
GATACCCTGAATACCTATCTTATAAGTTCCATATGGCTTATATTTCTAAGTAAGAGATGC
ACTTAGTAAGTGCATGTCGTCTTGACTTGTTTATACTCTAATGTATGATATTTATATCCC
TATAATATAGTGTTACTAATATATGTTTGGTATTGTGTAGACTCCATTGTACCATGGTGT
GCTAATTAGAAATAACATGCCAGCTTTGCTATTGTGGTTTGCAAGTAAAGTAAAAAAA
MacBook-Pro-de-Christine:Data tranchan$
```

“less” and “more” command

less <file_name> *writes the contents of a file one page at a time.*
less

Less Data/Fasta/EST-68566-Coffeacaneophora.fasta

```
MacBook-Pro-de-Christine:Data tranchan$ cat Data/Fasta/sequence.fasta
>Gxbjbsjxbjs
CCACCCCTCTTACAGTCTTCACCAAATGTCCTTTAAACTCCACCTAAAGTATCCAAAGA
CTCGAGAAATGCTGTGCCACAACCAGCTTTTGAGTCATCCATGACCGTTGATCTTCCTTT
GCCCCAGAGTGCGGCCTAGCACCATCTAGCTACTACTTGCCTTTCATACCCATCATTGG
GATACCCTGAATACCTATCTTATAAGTTCCATATGGCTTATATTTCTAAGTAAGAGATGC
ACTTAGTAAGTGCATGTCGTCTTGACTTGTTTATACTCTAATGTATGATATTTATATCCC
TATAATATAGTGTTACTAATATATGTTTGGTATTGTGTAGACTCCATTGTACCATGGTGT
GCTAATTAGAAATAACATGCCAGCTTTGCTATTGTGGTTTGCAAGTAAAGTAAAAAAA
MacBook-Pro-de-Christine:Data tranchan$
```

[space-bar] *to see another page*
[q] *to quit reading*
[/] *followed by the word*
to search

Up Down

more <file_name> *writes the contents of a file*

“nano” command

nano <file_name> *creates or edit a file called file_name*



The screenshot shows the nano text editor interface. At the top, a status bar displays "UW PICO 5.09" on the left and "File: draft.txt" on the right. The main editing area is a large, empty black rectangle. At the bottom, a command palette lists various shortcuts in two columns. The first column includes ^G Get Help and ^X Exit. The second column includes ^O WriteOut, ^J Justify, ^R Read File, ^W Where is, ^Y Prev Pg, ^V Next Pg, ^K Cut Text, ^U UnCut Text, ^C Cur Pos, and ^T To Spell. The ^W shortcut is highlighted with a red underline.

^G Get Help	^O WriteOut	^R Read File	^Y Prev Pg	^K Cut Text	^C Cur Pos
^X Exit	^J Justify	^W Where is	^V Next Pg	^U UnCut Text	^T To Spell



Practice

mkdir, cp, touch, cat

3

Go to [the working with files and directories](#) on
our website

Working with wildcard

Metacharacters : *, []

Why do we use the wildcards for?

- ✓ Can be used with all linux commands to increase the efficiency and flexibility of searches in Linux.

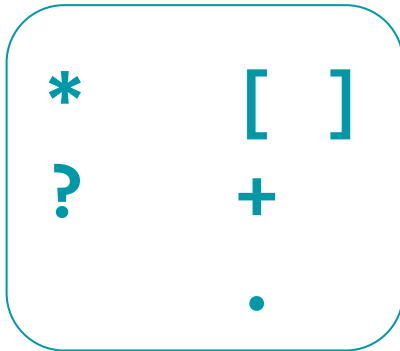


a symbol or set of symbols representing others characters

generally used as a substitute for any string or character

Why do we use the wildcards for?

✓ Allow to perform actions on more than one file at a time !



- to execute commands used to display the result
- to select part of files
- to find part of phrase in a file text
- many uses...

=> regular expressions to match the patterns

The "*" wildcard

Star wildcard matches one or more occurrences of any character, including no character.

*



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

How to list only the fastq files ?

The "*" wildcard

Star wildcard matches one or more occurrences of any character, including no character.

*



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

ls *fastq

The "*" wildcard

Star wildcard matches one or more occurrences of any character, including no character.

*



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

ls *fastq

KYVF-01.R1.fastq KYVF-02.R1.fastq Z016.fastq
KYVF-01.R2.fastq KYVF-02.R2.fastq

The "*" wildcard

Star wildcard matches one or more occurrences of any character, including no character.

*



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

ls KYVF*fastq

The "*" wildcard

Star wildcard matches one or more occurrences of any character, including no character.

*



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

ls KYVF*fastq

KYVF-01.R1.fastq KYVF-02.R1.fastq
KYVF-01.R2.fastq KYVF-02.R2.fastq

The “[]” wildcard

Square Brackets represent any of the characters enclosed in the brackets.



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

```
ls *. [sb]am
```

The “[]” wildcard

Square Brackets can represent any of the characters enclosed in the brackets.



```
KYVF-01.R1.fastq  
KYVF-01.R2.fastq
```

```
KYVF-02.R1.fastq  
KYVF-02.R2.fastq
```

```
KYVF.sam  
KYVF.bam
```

```
Z016.fastq  
Z016.bam
```

```
ls *.[sb]am
```

```
KYVF.sam  Z016.bam  
KYVF.bam
```

The “[]” wildcard

Square Brackets can represent any of the characters enclosed in the brackets.



KYVF-01.R1.fastq
KYVF-01.R2.fastq

KYVF-02.R1.fastq
KYVF-02.R2.fastq

KYVF.sam
KYVF.bam

Z016.fastq
Z016.bam

ls *. [sb]am

↔ ls *. [!f]*

KYVF.sam Z016.bam
KYVF.bam



Practice

`*, []`

4

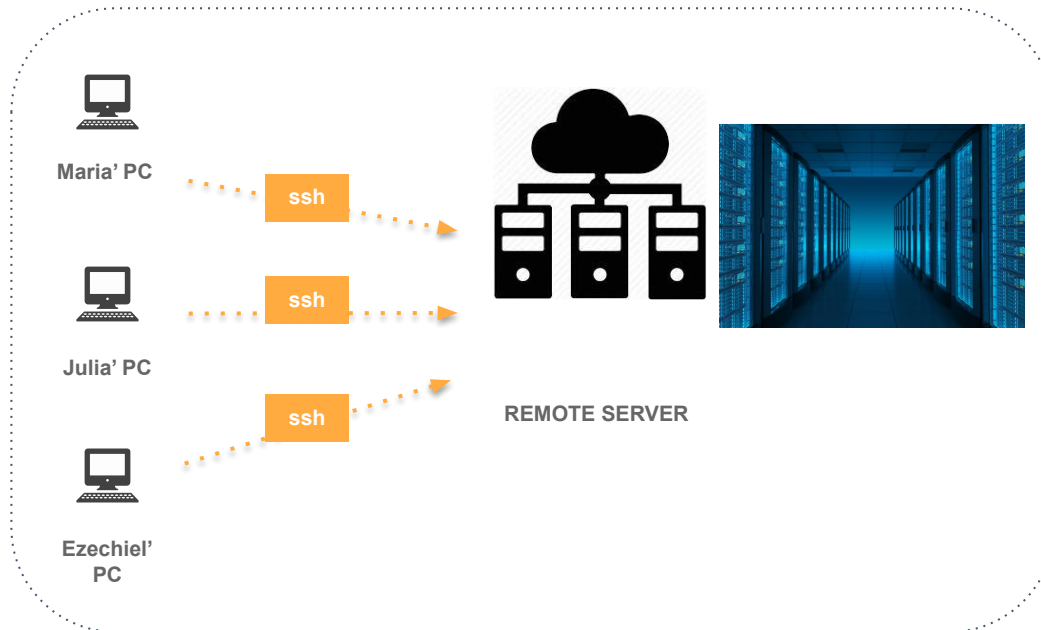
Go to [the wildcards practice](#) on our website

Remote server

How to work on a remote server?



Connect a remote server from your computer with the **ssh protocol**



SSH for **Secure Shell** protocol used to set up encrypted connections between two machines
=> a secure connection between your computer & the remote server, enabling you to work remotely

How to work on a remote server?



with the terminal
& ssh command



```
tranchan — CLUSTER — ssh bioinfo-inter.ird.fr -ltranchant — 130
Last login: Sat Mar 16 11:48:06 on ttys002
MacBook-Pro-de-Christine:~ tranchan$ ssh bioinfo-inter.ird.fr -ltranchant
Warning: Permanently added the ECDSA host key for IP address '64:ff9b::5bcb:2296'
Enter passphrase for key '/Users/tranchan/.ssh/id_rsa':
```



Where ?

*name of the remote
server*



Who ?

*account : Login &
pass*



Practice

ssh

5

Go to [the ssh practice](#) on our website

How to transfer files between your pc and a remote server?

Graphical interface

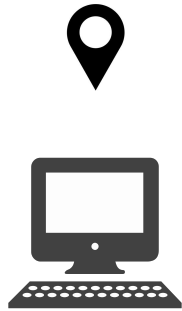


Command line

- “scp” command
- “rsync” command

How to use “scp” command?

```
scp -r <source> <destination>
```



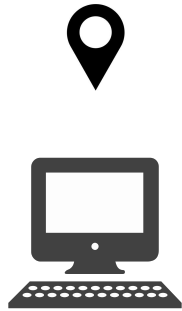
Your PC



server.ird.fr

How to use “scp” command?

```
scp -r <source> <destination>
```



Your PC

/home/user/



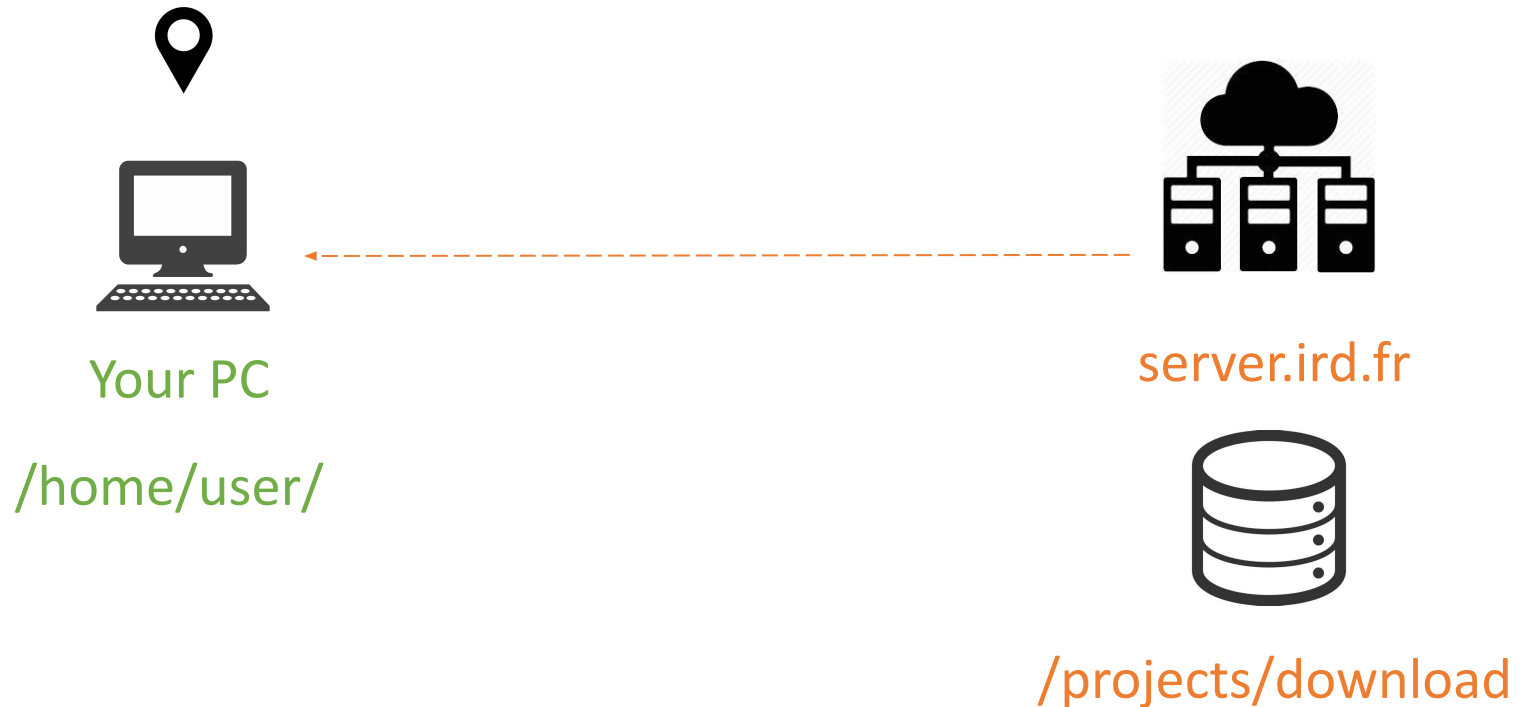
server.ird.fr



/projects/download

How to use "scp" command?

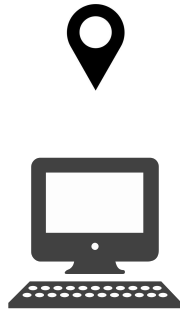
```
scp -r <source> <destination>
```



How to use "scp" command?

```
scp -r <source> <destination>
```

login:
formation



Your PC

/home/user/



server.ird.fr

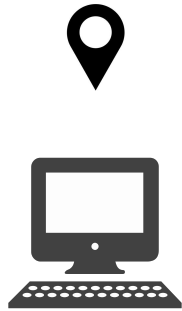


/projects/download

How to use "scp" command?

scp -r <source> <destination>

login:
formation



Your PC

/home/user/



server.ird.fr



/projects/download

connection

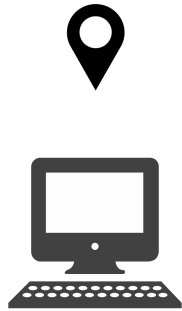
scp -r formation@server.ird.fr:/projects/download

source

How to use "scp" command?

scp -r <source> <destination>

login:
formation



Your PC

/home/user/



server.ird.fr



/projects/download

connection

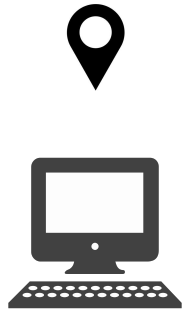
scp -r formation@server.ird.fr:/projects/download /home/user

source

destination

How to use “scp” command?

```
scp -r <source> <destination>
```



Your PC



/home/user/data



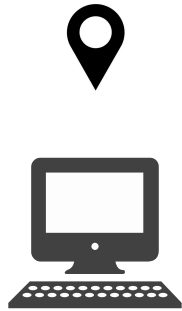
server.ird.fr

/projects/upload

How to use "scp" command?

```
scp -r <source> <destination>
```

login:
formation



Your PC



/home/user/data



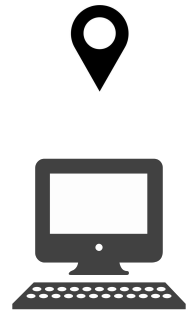
server.ird.fr

/projects/upload

How to use "scp" command?

scp -r <source> <destination>

login:
formation



Your PC



server.ird.fr

/projects/upload



/home/user/data

connection

scp -r /home/user/data formation@server.ird.fr:/projects/upload

source

destination

"wget" command

wget <file_url> *allows to retrieve a file from an url*

```
[dom@dom-XPS-13-9370 ~]$ wget https://github.com/lh3/bwa/archive/refs/tags/v0.7.18.tar.gz
--2024-11-18 10:49:30-- https://github.com/lh3/bwa/archive/refs/tags/v0.7.18.tar.gz
Résolution de github.com (github.com)... 140.82.121.4
Connexion à github.com (github.com)|140.82.121.4|:443... connecté.
requête HTTP transmise, en attente de la réponse... 302 Found
Emplacement : https://codeload.github.com/lh3/bwa/tar.gz/refs/tags/v0.7.18 [suivant]
--2024-11-18 10:49:31-- https://codeload.github.com/lh3/bwa/tar.gz/refs/tags/v0.7.18
Résolution de codeload.github.com (codeload.github.com)... 140.82.121.9
Connexion à codeload.github.com (codeload.github.com)|140.82.121.9|:443... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Taille : non indiqué [application/x-gzip]
Enregistre : 'v0.7.18.tar.gz'

v0.7.18.tar.gz [ <=>

2024-11-18 10:49:31 (3,19 MB/s) - 'v0.7.18.tar.gz' enregistré [238036]
```

Manipulating compressed files

Compressing files: **tar, gzip**

```
tar -zcvf tarfile.tar.gz dirToCompress
gzip fileToCompress
```

Decompressing archives: **gunzip, tar**

```
gunzip file.gz
tar -xvf file.tar
tar -zxvf file.tar.gz
gzip -d file.gz
```

Displaying the contents of an archive: **zcat**

```
zcat data.txt.gz
```

Searching for an expression/pattern in a compressed file: **zgrep**

```
zgrep 'NM_000020' data.gz
```



Practice

filezilla,scp

6

Go to [the filezilla, scp practice](#) on our website

Search patterns and manipulate files

Useful commands

head writes the first ten lines of a file to the screen

```
head -n 20  
script.pl
```

tail writes the last ten lines of a file to the screen

```
tail -n 5 script.pl
```

wc Count for word, lines, characters in a file

```
wc script.pl  
wc -l script.pl
```

"sort" command

sort

sort the content of a text file, line by line

sort -t SEPARATOR -k ... fileName

<i>sort -k2 fileName</i>	Alphabetical sorting based on the second column
<i>sort -k2r fileName</i>	Reverse Alphabetical sorting based on the 2nd col
<i>sort -t: -k3g fileName</i>	-t option defines the field separator (by default :)
<i>sort -k2g -k1r fileName</i>	Numeral sorting on the 2nd column then the 1st column

"cut" command

cut

Extracts columns/fields from a file

cut -d SEPARATOR -f fieldsNumber fileName

“cut” command

cut

Extracts columns/fields from a file

cut -d SEPARATOR -f fieldsNumber fileName

cut -d “:” -f1,5 /etc/passwd



Picked up the FIRST and FIFTH columns of FILE,
separated by :

"grep" command

grep

searching a word, a pattern in a file

grep [options] pattern_searched file1 file2

"grep" command

grep

searching a word, a pattern in a file

grep [options] pattern_searched file1 file2

*grep "ATTCG"
allSeq.fasta*

"grep" command

grep

searching a word, a pattern in a file

grep [options] pattern_searched file1 file2

grep ">" allSeq.fasta



**Don't forget to enclose it
with single/double quotes**



Practice

head,tail,cut, grep

7

Go to [the manipulation files practice](#) on our
website

"sed" command

Substitution/Replacement in lines

Select lines in a file using a regular expression
AND apply a modification o/treatment to these lines

```
sed "s/pattern search/new pattern/" file
```

substitution

separator

searched pattern

new pattern

file to parse

“sed” command

A few examples

Example	Description
<code>sed "s/day/night/" file</code>	Change the 1st occurrence of “day” by “night” per line
<code>sed "s/[lL]inux/LINUX/g" file</code>	Change all occurrences of “linux” by “LINUX”

Input/output redirection

To save the output of a command in a file

Output redirection

✓ Redirection ?

to save the output of a command in a file instead of printing it on our terminal

simply use the “>” character

command 1 > file_path

Output redirection: use it with caution



✓ Overwrite Redirection >

Replace all the existing content of that file ⇔
the file will be overwritten and will contain only the output
of the redirected command

```
cut -d: -f1 /etc/passwd > userName.txt
```

✓ Append Redirection >>

To add few lines to the end of an existing file.

```
cut -d: -f1 /etc/passwd >> userName.txt
```

Chaining commands

Chaining commands

The standard output of a first command can be sent as standard input to another command with the **| operator**

**To connect several commands together
(without using temporary files)**

```
cmd1 | cmd2 | cmd3
```

Chaining commands

; command2 will be executed regardless of whether command1 has been executed successfully or not

```
cd ~/LINUX_TP; ls right  
cd ~/LIIINUX_TP; ls wrong
```

&& command2 will execute if command1 has been executed successfully.

```
cd ~/LIIINUX_TP && ls
```

Chaining commands: example

```
cut -d: -f1 /etc/passwd
```

```
Root
```

```
troot
```

```
iroot
```

```
ctroot
```

```
//
```

Chaining commands: example

```
cut -d: -f1 /etc/passwd
```

```
Root  
troot  
iroot  
ctroot  
"
```

```
cut -d: -f1 /etc/passwd | sort
```

```
abate  
adm  
adroot  
ais  
#albar  
alvaro-wis  
anthony  
apache
```


Chaining commands: example

```
cut -d: -f1 /etc/passwd
```

```
Root  
troot  
iroot  
ctroot  
"
```

```
cut -d: -f1 /etc/passwd | sort
```

```
abate  
adm  
adroot  
ais  
#albar  
alvaro-wis  
anthony  
apache
```

```
cut -d: -f1 /etc/passwd | sort | head
```

Chaining commands: example

```
cut -d: -f1 /etc/passwd
```

```
Root  
troot  
iroot  
ctroot  
"
```

```
cut -d: -f1 /etc/passwd | sort
```

```
abate  
adm  
adroot  
ais  
#albar  
alvaro-wis  
anthony  
apache
```

```
cut -d: -f1 /etc/passwd | sort | head > /etc/passwd.sort
```



Practice

>, >>, |, &&, ;

8

Go to [the chaining commands practice](#) on our
website

Scripting in bash

What is a bash script?

- A way to group your command in a single file
- Commands are executed in an sequential mode
- Allows you to automate your work

Rules to create a bash script

- Always start with : `#!/bin/sh`
- One instruction per line
- **Each instruction should end with ;**
- Use the `#` to comment your script

Rules to create a bash script

- Always start with : `#!/bin/sh`
- One instruction per line
- **Each instruction should end with ;**
- Use the `#` to comment your script:
 - script ignore what is after the `#`
 - Add infos for you and your colleagues

Additional Tip

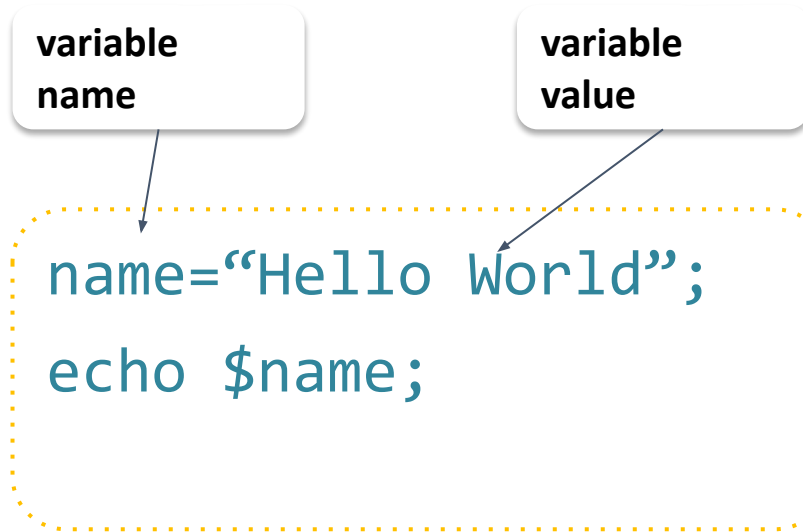
- Use “echo” command to display informations along your script:

`echo 'text';` To display on output (screen)

`echo -e “text \n”;` To go the line after

Variables

Variable...



A box into which you can store an object or a information.

Rules

- Variable names only with *alpha-numeric characters* (A-Z, a-z, 0-9) or *underscore*
- **Case sensitive , non space in the name!**

variables : results from command

- To store the value of a command, use the following syntax `$(command)`

```
result_command=$(ls /home/user);  
echo $result_command;
```

Execute a bash script

- Make it executable:

```
chmod +x <script name>
```

- Execute the script:

```
sh <script name>
```

- Execute the script with arguments :

```
sh <script name> arg1 arg2
```

bash script arguments

```
sh <script name> arg1 arg2
```

- Used to give input values at the runtime for the script
- No need for hard coded values in the script
- scripts more dynamic and reusable
- The script uses special variables:
 - `$0` : the script itself
 - `$1` : first argument, `$2` : 2nd argument etc
 - `$#`: number of arguments passed to the script
 - `$@`: contains all the input arguments

example bash script arguments

```
sh example .sh arg1 arg2 arg3
```

- \$0 : example.sh
- \$1 :arg1, \$2 : arg2, \$3: arg3
- \$#: 3
- \$@: arg1 arg2 arg3



Practice

sh

9

Go to [scripting practice](#) on our website

Conditions

"if" loop

if... **if** TEST-COMMAND

then

STATEMENT 1

fi

If the variable condition against the value is true then the action is executed

```
if [[ variable condition value ]]  
then  
    instruction1  
    instruction2  
else  
    instruction3  
fi
```


"if" loop

if... **if** TEST-COMMAND

 then

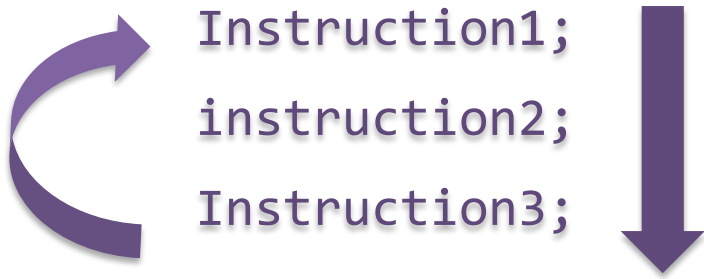
 STATEMENT 1
 else
 STATEMENT 2
 fi

If the variable condition against the value is true then the action is executed **otherwise the else condition is applied**

```
if [[ variable condition value ]]  
then  
    instruction1  
    instruction2  
else  
    instruction3  
fi
```

“for” loop

for...



- To parse a directory
- To run the same instruction on each file of the directory

```
for file in * ;  
do  
    instruction1  
    instruction2  
done
```



Practice

sh

10

*Go to [the 2nd scripting practice](#) on our
website*

Other useful commands

Disk space and file size

Disk (free) size: **df**

disk free

`df`

occupied space in bytes

`df -h`

human-readable

Directory size: **du**

disk usage

`du`

`du -h`

`du -h *`

Searching for a file by its name **find**

`find -name "transcriptsAssembly.fasta"`

“history” command

history

displays all the last commands that have been executed in all the previous sessions

The entire history is saved into the file `.bash_history`

```
tranchan — tranchant@master0:~ — ssh bioinfo-master.ird.fr -ltranchant — 93x22
...nt@master0:~ — ssh bioinfo-master.ird.fr -ltranchant  ~ — tranchant@master0:~ — -bash  ...  ~ — CLUSTER — -bash
[[tranchant@master0 ~]$ history | head
 23  sh nucmer.sh
 24  qstat
 25  qrsh
 26  cd /data3/projects/africanRice/
 27  cd Abyss/NucmerAlignement/AA/individualAlignment/
 28  ls
 29  vi nucmer.sh
 30  ll
 31  ls
 32  pwd
[tranchant@master0 ~]$
```

SouthGreen
bioinformatics platform

displays all the last commands that have

“history” command

history

displays all the last commands that have been executed in all the previous sessions

The entire history is saved into the file `.bash_history`

Filtering the History Output

```
history | grep "blastn"
```

displays only the commands including the search keyword “blastn”

```
history | tail
```

displays the commands recently used

```
history | grep "blastn" | tail -n 5
```

```
history | head -n 5
```

displays the oldest commands

renaming files

rename

Example	Description
<code>rename 's/.txt/.fasta/' *</code>	rename the extension of all files
<code>rename 'y/a-z/A-Z/' *</code>	rename files in uppercase

Thank you for your attention !



Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International:

<http://creativecommons.org/licenses/by-nc-sa/4.0/>

awk

awk: Language for manipulating a tabulated file line by line

- Authors' names : “Aho, Weinberger, and Kernighan”
- A programming language that makes it easy to manipulate tabulated files (blast, sam, vcf) and extract part of the data.
- A language used to search for patterns and perform associated operations and actions.

awk

Syntax : awk [-F] 'program' file

Option	Description
-F	specifies field separators

awk

Syntax : awk [-F] 'program' file

Option	Description
-F	specifies field separators

Predefined variables used by awk

Variable	Description
\$0	Line
NR	Line number read
NF	Number of fields in line

awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcaron@videotron.ca

File: contact.txt

awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcaron@videotron.ca

File: contact.txt

```
awk '{print $0}' contact.txt
```

```
Helene 56 edu hcyr@sun.com  
jean 32 ri jeanc@inexpress.net  
julie 22 adm juliem@sympatico.ca  
michel 24 inf michel@uqo.ca  
richard 25 inf rcaron@videotron.ca
```

print each line
read

awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcaron@videotron.ca

File: contact.txt

```
$awk '{print NR, $1, $2}' contact.txt
```

```
1 Helene 56
2 jean 32
3 julie 22
4 michel 24
5 richard 25
```

Displays the number of the line read
then the 1^{rst} field
then the 2^e field of the tabulated file

awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcaron@videotron.ca

```
$awk '{print $1,$2};  
END { print NR "lines read in the file" }' contact.txt
```

Helene 56

Jean 32

Julie 22

Michel 24

Richard 25

5 lines read in the file

Instruction executed at
the end of file reading

awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcaron@videotron.ca

```
$awk '{print $1,$3; sum+=$2}  
END { print "Sum of all ages equal to ", sum }' contact.txt
```

Helene edu

jean ri

julie adm

michel inf

richard inf

Sum of all ages equal to 159

We add the age (\$2) to the **sum** variable for each line read.

Then we **display the sum** calculated at the end of file reading

awk

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcaron@videotron.ca

File: contact.txt

```
$awk ' {sum+=$2}  
END { print " Mean age = ", sum/NR } ' contact.txt
```

Mean age = 31,8

We add the age (\$2) to the **sum** variable for each line read

Then we display the **mean age** once the file has been read

awk

we can apply conditions!

if (Condition) {Instr-1; Instr-2; ...; Instr-n}

With 2
conditions



```
awk '{if ( $2 > 24 && $2 < 50 ) { print $1 " 's age between  
24 and 50 : equal to ", $2 } }' contact.txt
```

```
Helene's age between 24 and 50 : equal to 56  
jean's age between 24 and 50 : equal to 32  
richard's age between 24 and 50 : equal to 25
```

awk

```
awk '{if ($3 == "inf") {print $0} }' contact.txt
```

```
michel 24 inf michel@uqo.ca  
richard 25 inf rcaron@videotron.ca
```

```
awk '/j/ {print $0}' contact.txt
```

```
jean 32 ri jeanc@inexpress.net  
julie 22 adm juliem@sympatico.ca
```

awk

```
awk '{print $1, $2-10}' contact.txt
```

```
Helene 46  
Jean 12  
Julie 12  
Michel 14  
Richard 15
```

```
awk '{if($2 > 30 && $3 == "ri") {print $0}}' contact.txt
```

```
jean 32 ri jeanc@inexpress.net
```

These commands can be used with standard output or a tabulated file (such as .gff, blast m8 file, .vcf) as input.

File attributes and permissions

Command: `ls -l`

```
$ ls -l filename  
drwxrwxrwx 3 user user 4096 2012-02-11 20:21 file_name
```

Permissions

Owner

Group

Size

Last modification date and time

Permission legend / interpretation

Type

- : standard file

d : directory

l : symbolic link

File attributes and permissions

ls -l command

Permissions

`drwxrwxrwx 3 user user 4096 2012-02-11 20:21 file_name`

other
group
user

3 classes


3 types of permissions :

Permission	File	Directory
Read r	Open and read	List and et copy files
Write w	Modify and remove	Manipulate contents: copy, create, modify, overwrite
Execution x	Execute file	Access to contained files for execution

File attributes and permissions

permission management command: **chmod**

```
chmod <perm> file_name
```



Each permission = 1 value

R	4
W	2
X	1
none	0

Example

```
chmod 740 script.sh
```

```
chmod 755 script.sh
```

```
# Owner=rwx Group=r-- Other=---
```

```
# Owner=rwx Group=r-x Other=r-x
```


Visualize and modify permissions

chmod, ls

Provide owner name, group name and permissions for files contained in directory “~/Data/454-projet1/raw”

Modify permissions on file Scripts/blast.pl to set them as follows:
read and write for the group
read, write, execute for the owner
read for others (public)

