



Analyse de données de séquençage GBS

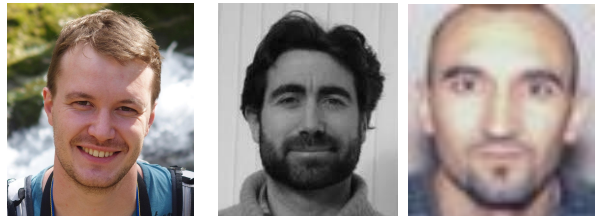
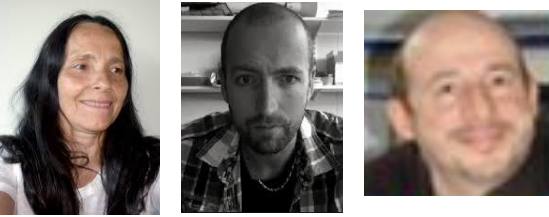
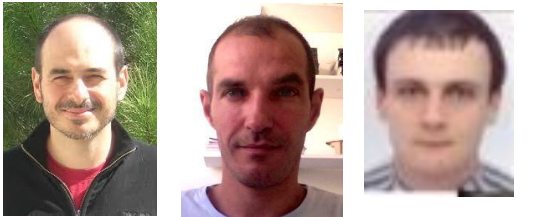
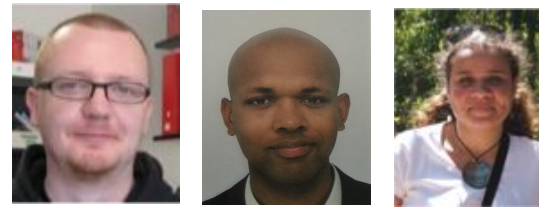
Formation bioinformatique

Guadeloupe - 28 Janvier au 1^{er} Février 2019

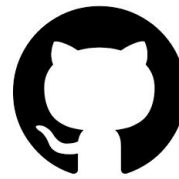
<https://southgreenplatform.github.io/trainings/gbsGuadeloupe/>

Programme de la formation

28 Janvier	Initiation à Linux
29 et 30 Janvier	Détection de Variants
31 Janvier (matin)	Initiation à Galaxy
31 Janvier et 1er Février	Reconstruction de génomes mosaïques



www.southgreen.fr



<https://github.com/SouthGreenPlatform>

southgreenplatform.github.io/trainings/

southgreenplatform.github.io/tutorials/



The South Green portal: a comprehensive resource for tropical and Mediterranean crop genomics, Current Plant Biology, 2016

Guide de survie à Linux

Guadeloupe - 28 Janvier au 1^{er} Février

Gautier Sarah – gautier.sarah@inra.fr

d'après un support de

Christine Tranchant – christine.tranchant@ird.fr

Ndomassi Tando – ndomassi.tando@ird.fr

Bertrand Pitollat – bertrand.pitollat@cirad.fr



The objectif!

Lancez vos analyses sous Linux !



Après ce module, vous serez capable de :

- Connaître les principales commandes Linux
- Se déplacer dans l'arborescence de fichier : *pwd*, *ls*, *cd*, *mkdir* etc.
- Se connecter à un serveur, transférer les données : *ssh*, *scp*, *wget*
- Manipuler des fichiers : *head*, *tail*, *sort*, *cut*, *wc*
- Lancer des logiciels en ligne de commande



Introduction

- **Systeme d'exploitation** réputé pour :
 - sa sécurité
 - ses mises à jour fréquentes
 - son prix et ses programmes gratuits
- Créé en 1991 par **Linus Torvalds**
- Basé sur l'OS propriétaire Unix (1969)
- Code source de linux **gratuit** et **libre** : copier, modifier, redistribuer



- **OS robuste et multi-plateforme**

(ordinateur, serveur, android...)



- **Système multi-utilisateurs**

Plusieurs utilisateurs peuvent travailler en même temps

- **Système multi-tâches (processus/programmes)**

Chaque utilisateur peut lancer plusieurs programmes en même temps

- 2 façons d'utiliser linux :

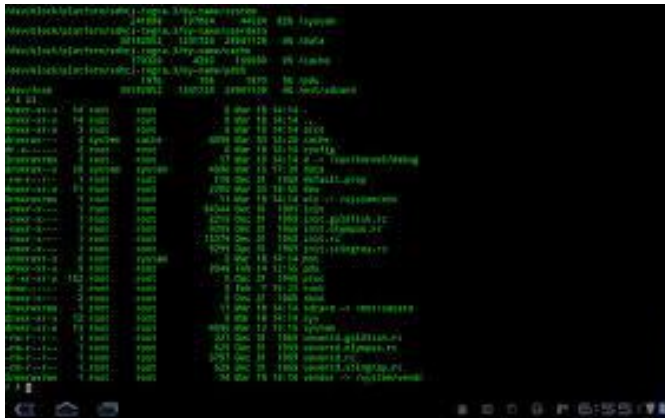
en *mode graphique*



- 2 façons d'utiliser linux :

en *mode graphique*

en *mode console* (terminal)

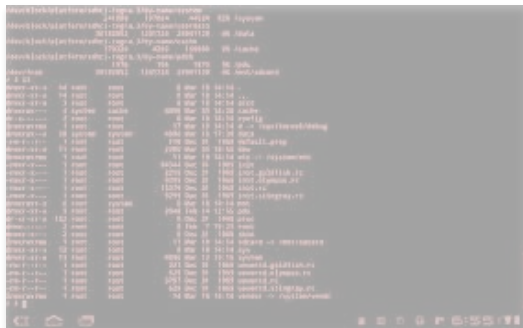




- Nombreux programmes rapides & puissants
- Facile de lier des commandes/programmes entre eux (workflow)
- Nombreux outils bioinformatique disponibles
- Pas besoin de ressources matérielles importantes
- 90% des serveurs fonctionnent sous Linux



- Nombreux programmes rapides & puissants
- Facile de lier des commandes/programmes entre eux (workflow)
- Nombreux outils bioinformatique disponibles
- Pas besoin de ressources matérielles importantes
- 90% des serveurs fonctionnent sous Linux



**Pas d'interfaces
graphiques**

**Convivialité de la ligne
de commande ?**





Nécessité de la pratique et de l'expérience

⇔ Investissement non négligeable pour de bons résultats rapidement

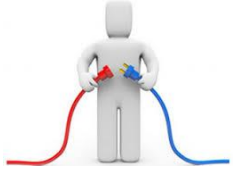


Environnement de travail

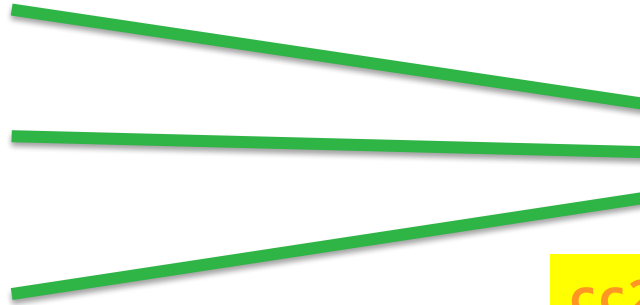
*Comment copier un fichier de son PC sur
le serveur ?*



Copier un fichier de son PC sur le serveur ?



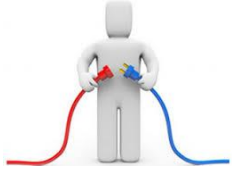
- En se connectant sur un serveur linux distant de son portable windows ou mac via le *protocole sftp*



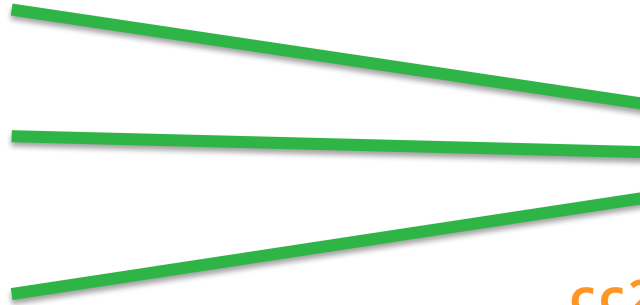
cc2-login.cirad.fr



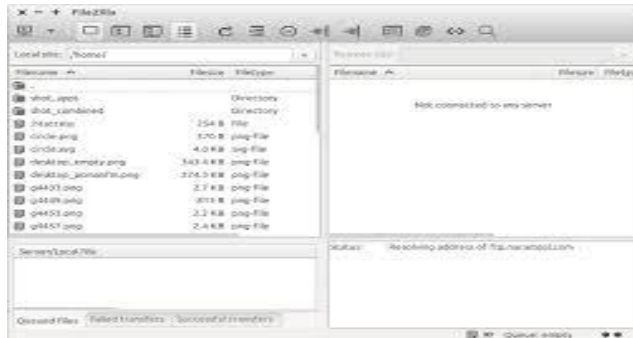
Copier un fichier de son PC sur le serveur ?



- En se connectant sur un serveur linux distant de son portable windows ou mac via le **protocole sftp**



cc2-login.cirad.fr





Practice

filezilla, sftp

1

Go to [Practice 1](#) on our github

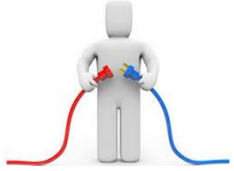


Environnement de travail

Comment travailler sur le serveur ?



Comment travailler sur le serveur ?

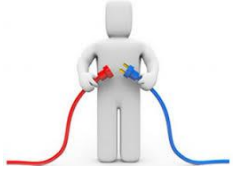


- En se connectant sur un serveur linux distant de son portable windows ou mac via le *protocole ssh*

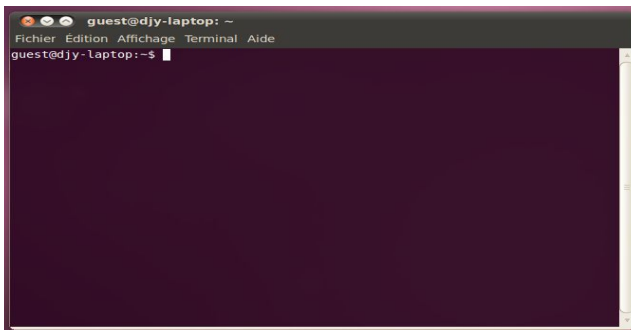
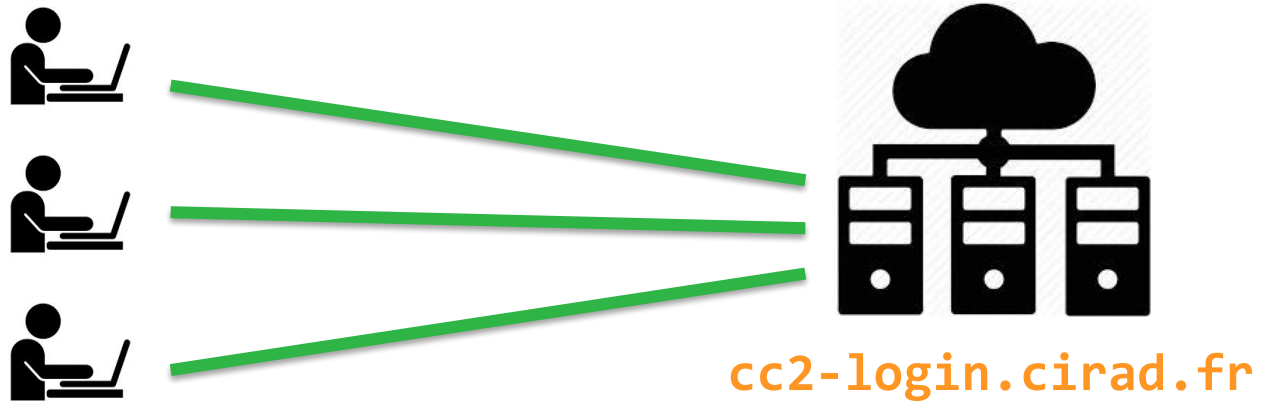




Comment travailler sur le serveur ?



- En se connectant sur un serveur linux distant de son portable windows ou mac via le *protocole ssh*





Practice

putty,
terminal, ssh

2

Go to [Practice 2](#) on our github



Premiers Pas sur Linux

Se déplacer dans l'arborescence de fichiers et manipuler des fichiers/répertoires

Toujours présent sur le terminal, juste avant de taper la commande

Prompt

```
[tranchant@node6 data]$
```


Toujours présent sur le terminal, juste avant de taper la commande

Prompt

```
[tranchant@node6 data]$
```

Nom
utilisateur

Nom
serveur

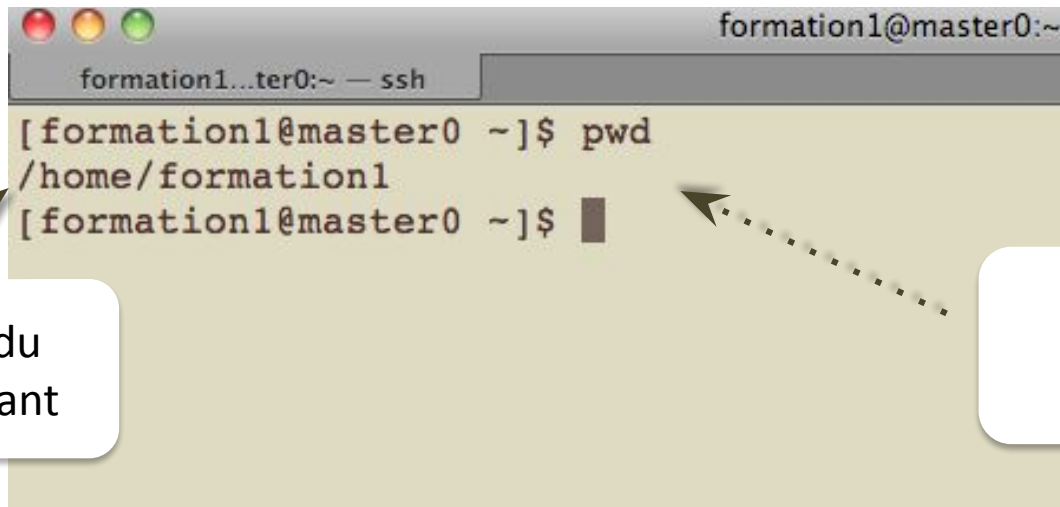
Répertoire
courant

commande [-options] [arguments]

pwd

Present Work Directory

affiche nom du répertoire courant (et son chemin complet)



```
formation1@master0:~  
formation1...ter0:~ — ssh  
[formation1@master0 ~]$ pwd  
/home/formation1  
[formation1@master0 ~]$ █
```

Affiche le nom du répertoire courant

Commande sans options et sans argument

ls
list

liste fichiers & répertoires présents dans le répertoire courant

```
formation1@master0:~ — ssh — 97x37  
formation1@master0:~ — ssh  
[formation1@master0 ~]$ ls  
data  scripts
```

Liste des fichiers du répertoire courant (par défaut)

Commande sans options et sans argument

ls -l
list long

liste les fichiers avec des informations pour chaque fichier/répertoire

Commande avec l'option **-l** et le **nom d'un répertoire** en 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 ayouba    team1        4096 13 avril  2012 ayouba
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
  
```

liste détaillée des fichiers

Comment obtenir de l'aide sur une commande?

- avec l'option *--help* ou *-h*

ls --help

blastn -h

- avec la commande *man*

man ls

Arborescence linux

- pwd** Affiche le chemin absolu
- ls** Liste tous les fichiers/répertoires
- ls -l** Affiche toutes les informations sur les fichiers



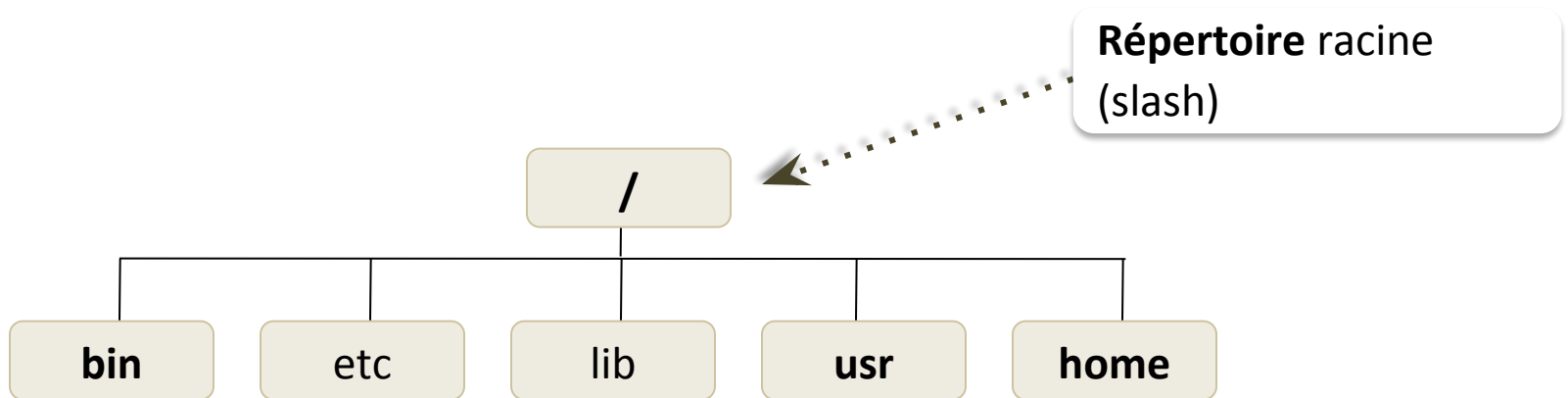
Practice

prompt, pwd

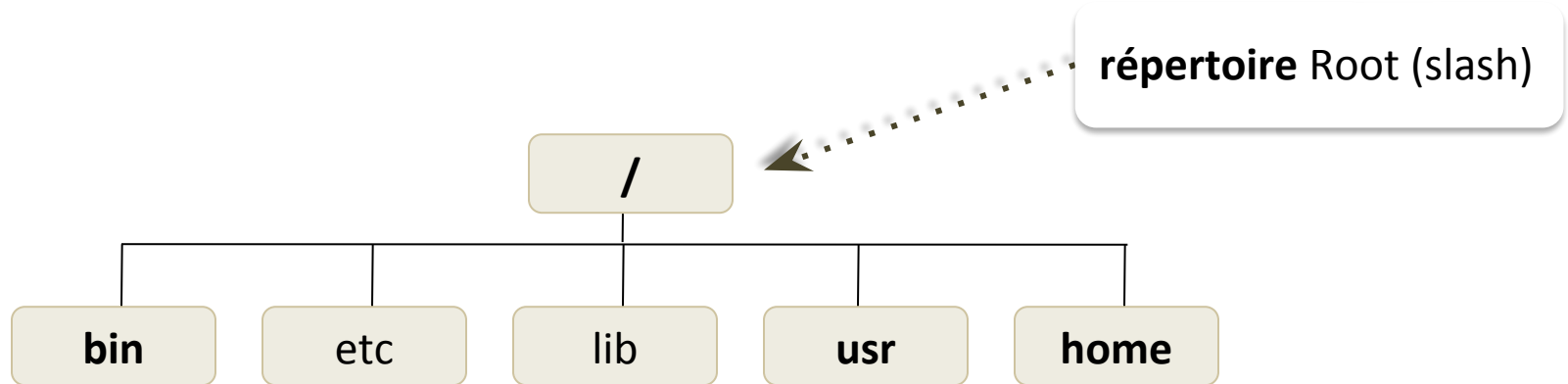
3

Go to [Practice 3](#) on our github

- Arborescence = système de fichiers
- Répertoire racine appelé “/”



Principaux répertoires



/bin **commandes principales**

/usr, /opt **Applications et librairies**

/usr/bin **Autres commandes**

/home **répertoire des utilisateurs (1 par utilisateur/login)**

Chemin (path) : chemin d'accès à un fichier/répertoire

Chemin (path) : chemin d'accès à un fichier/répertoire

absolu

- chemin complet du fichier en partant du répertoire racine /

Chemin (path) : chemin d'accès à un fichier/répertoire

absolu

- chemin complet du fichier en partant du répertoire racine /
- *commence toujours par /*
- **toujours correct, peu importe où l'on travaille**

Chemin (path) : chemin d'accès à un fichier/répertoire

absolu

- chemin complet du fichier en partant du répertoire racine /
- commence toujours par /
- toujours correct peu importe où l'on travaille

relatif

- chemin défini par rapport où on est dans l'arborescence

Chemin (path) : chemin d'accès à un fichier/répertoire

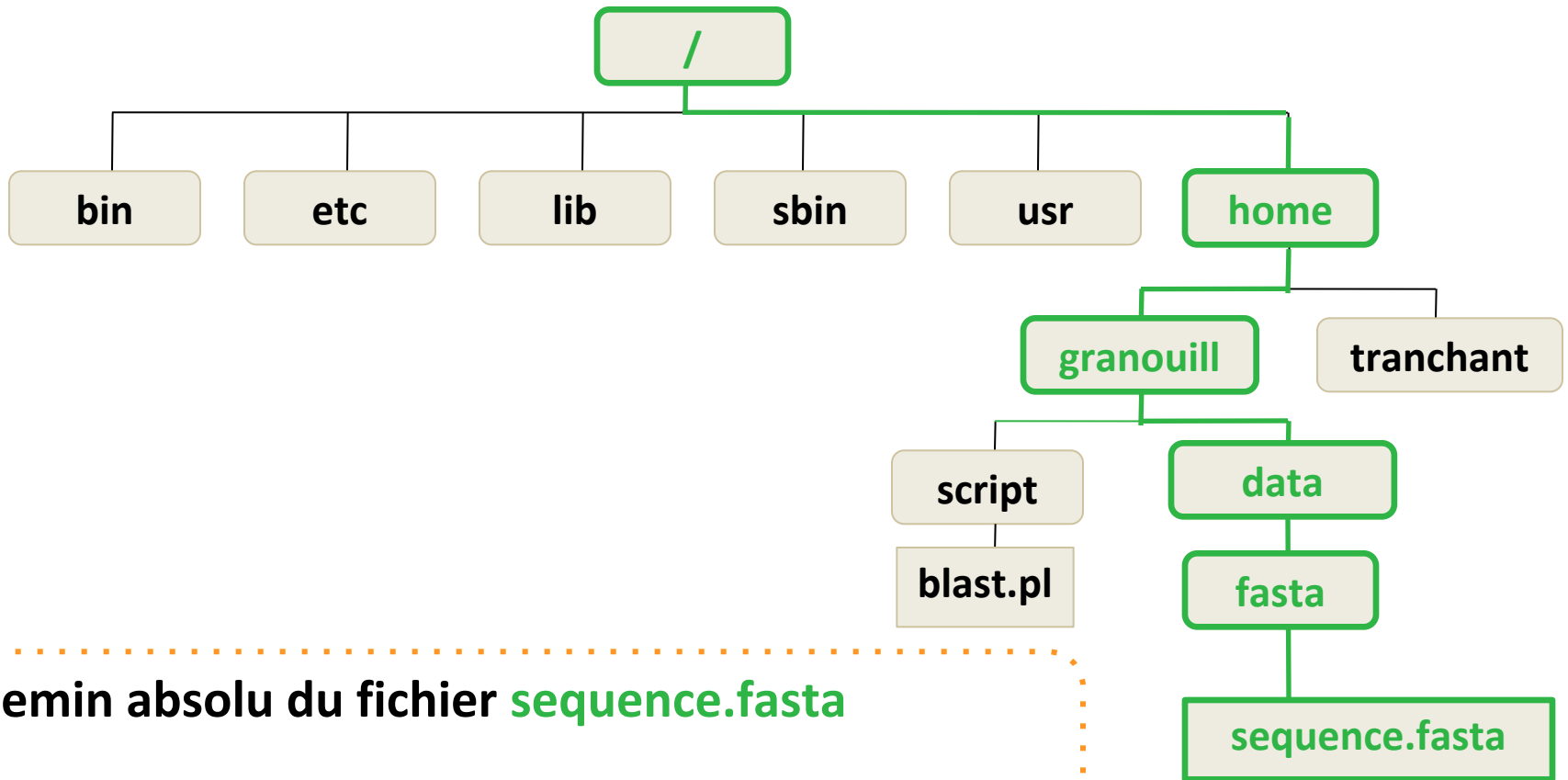
absolu

- chemin complet du fichier en partant du répertoire racine /
- commence toujours par /
- toujours correct peu importe où l'on travaille

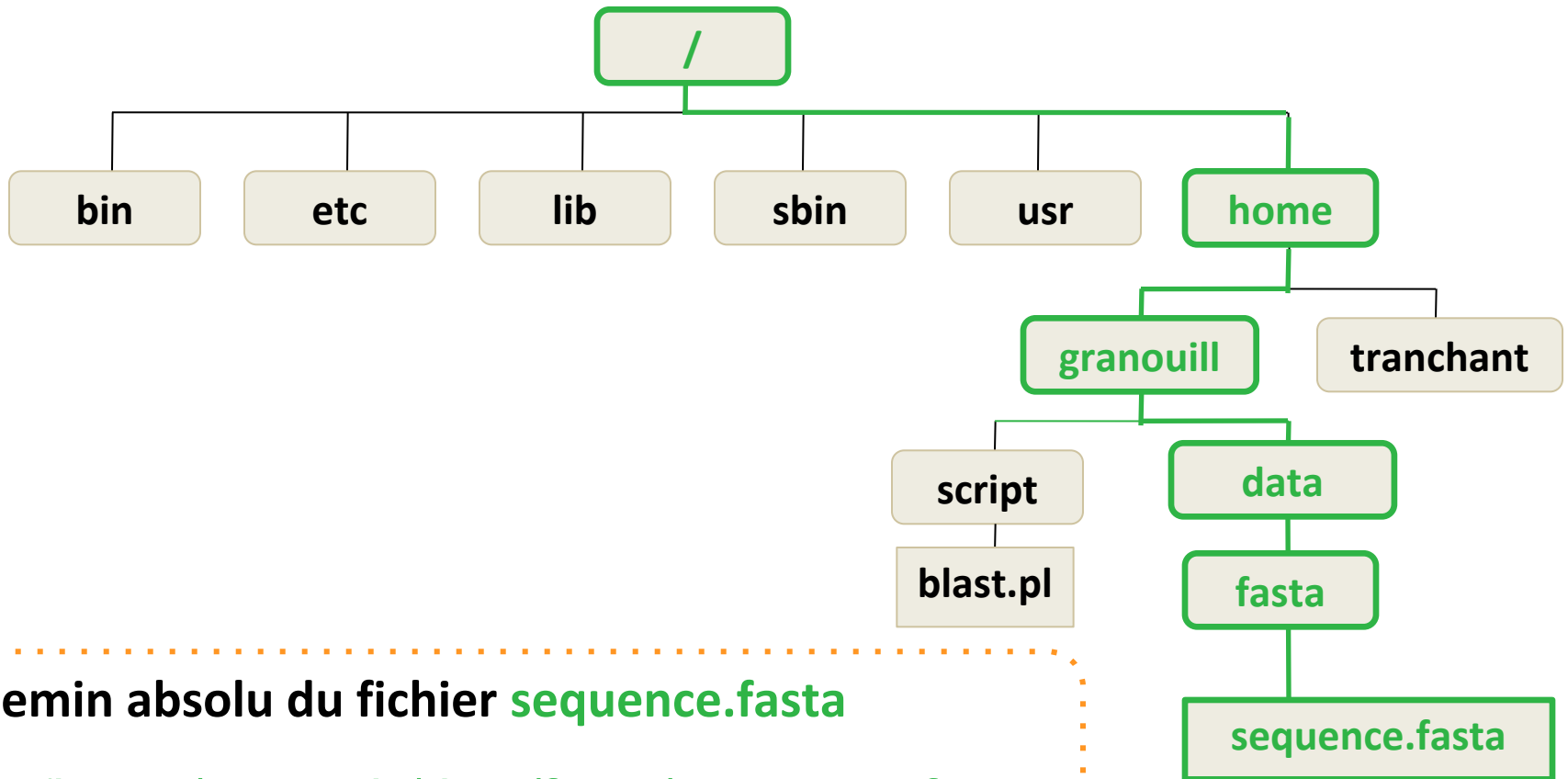
relatif

- chemin défini par rapport où on est dans l'arborescence
- *ne commence jamais par /*
- **change selon où l'on travaille**

- Commence toujours par / (le répertoire racine)
- **Toujours correct peu importe où l'on travaille**



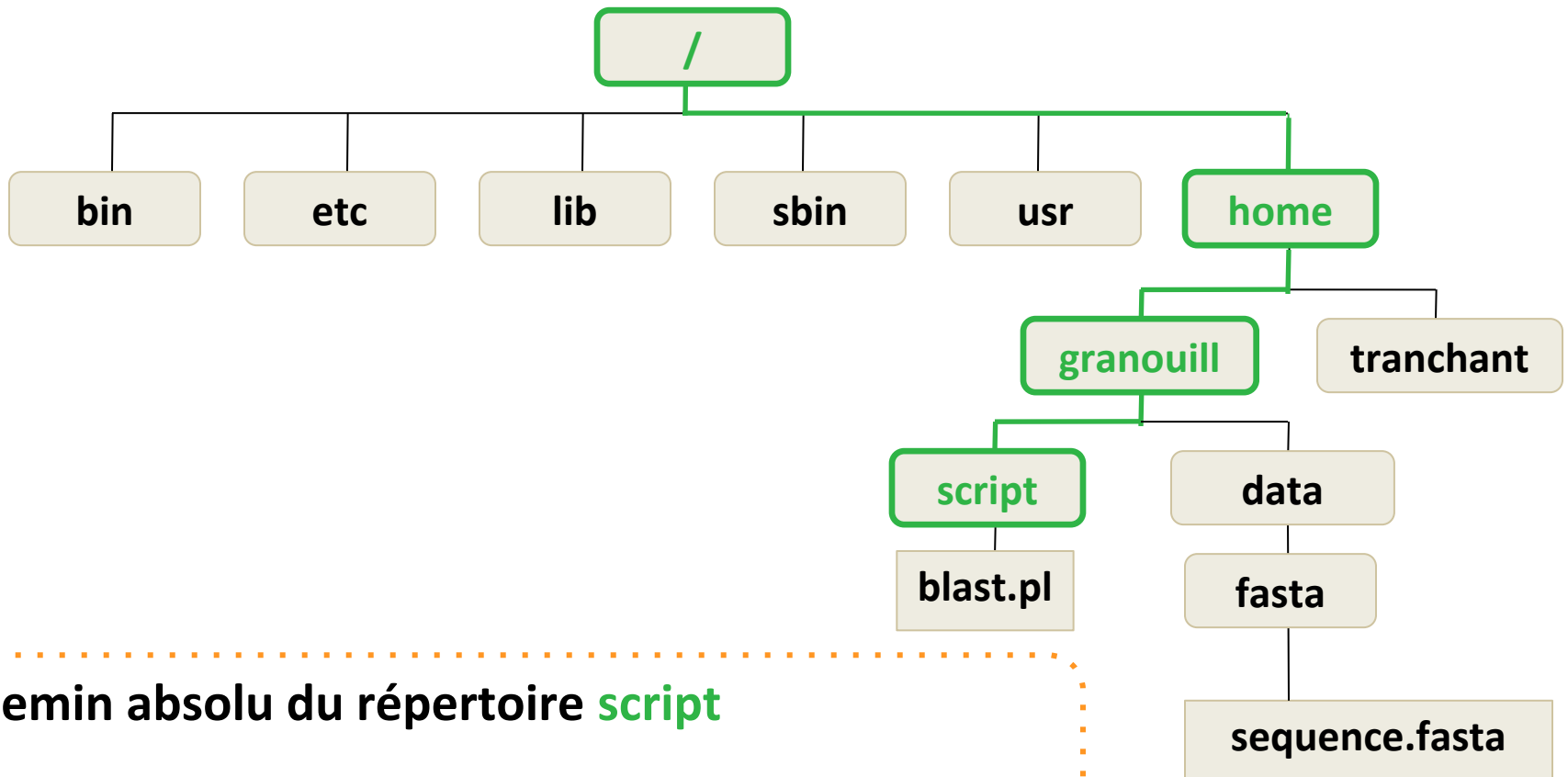
- Commence toujours par / (le répertoire racine)
- **Toujours correct peu importe où l'on travaille**



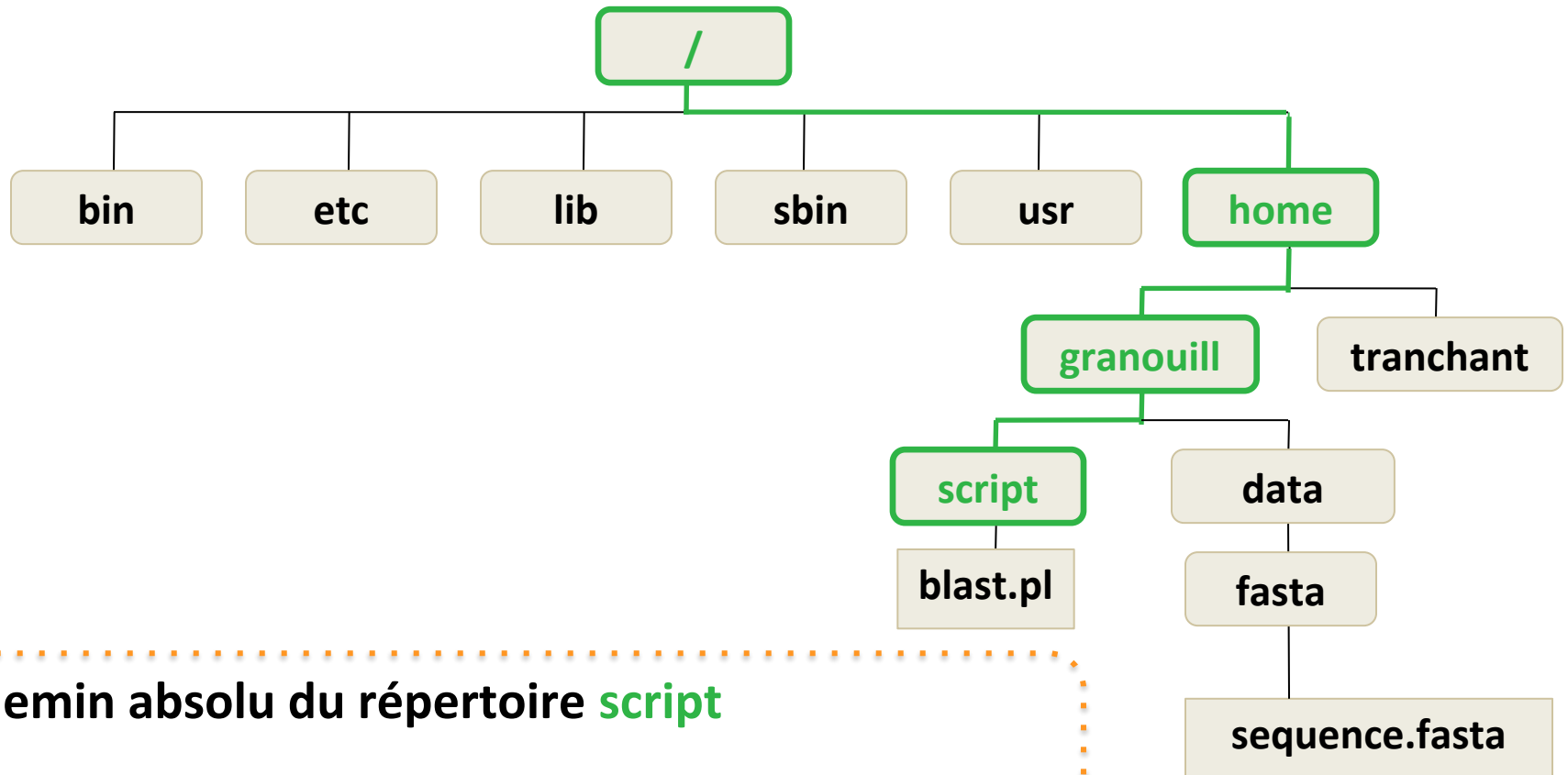
Chemin absolu du fichier **sequence.fasta**

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

- Commence toujours par / (le répertoire racine)
- **Toujours correct peu importe où l'on travaille**



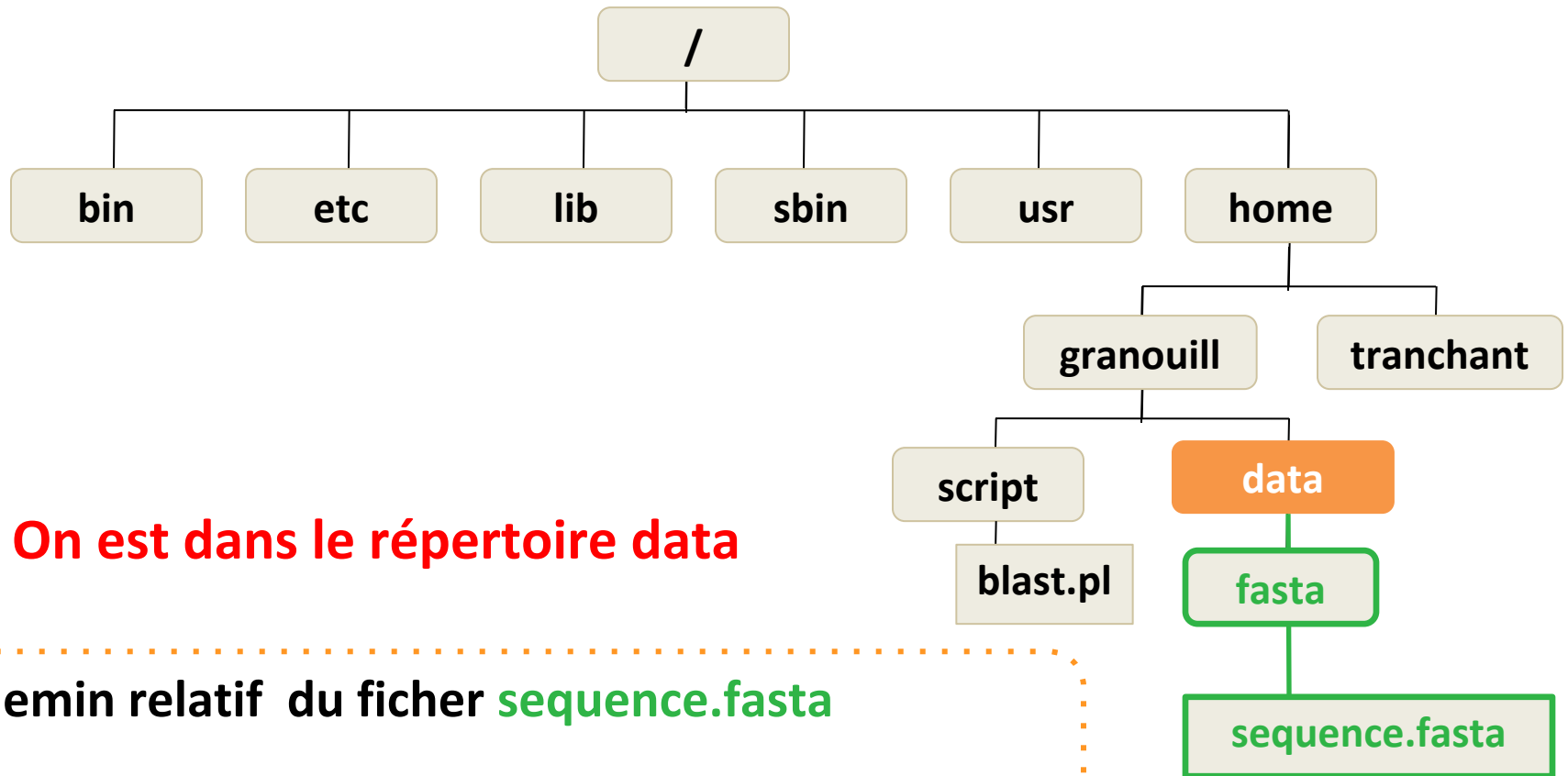
- Commence toujours par / (le répertoire racine)
- **Toujours correct peu importe où l'on travaille**



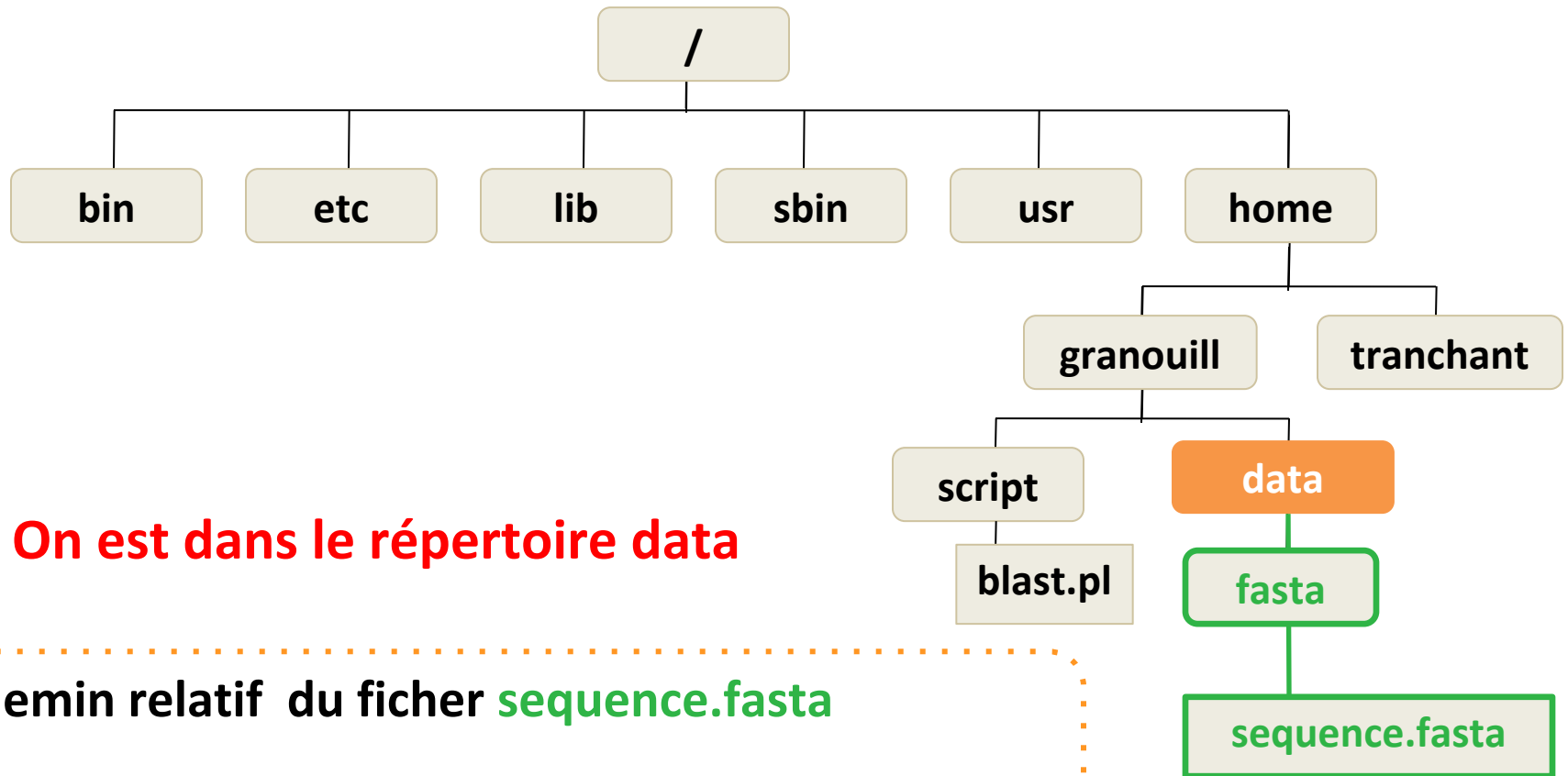
Chemin absolu du répertoire **script**

`/home/granouill/data/script`

- Défini par rapport où on est dans l'arborescence
- **Ne commence jamais par /**



- Défini par rapport où on est dans l'arborescence
- **Ne commence jamais par /**

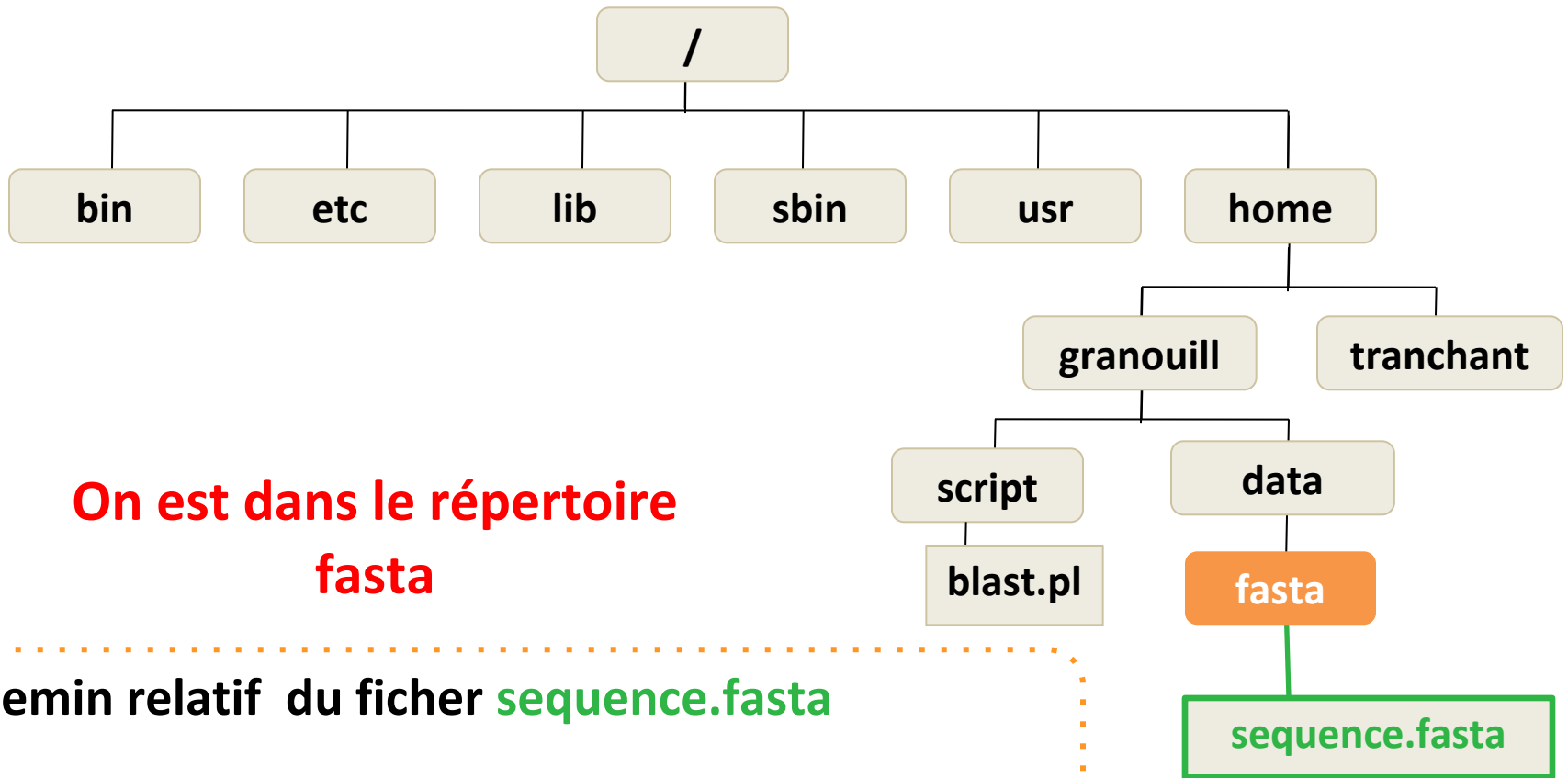


On est dans le répertoire data

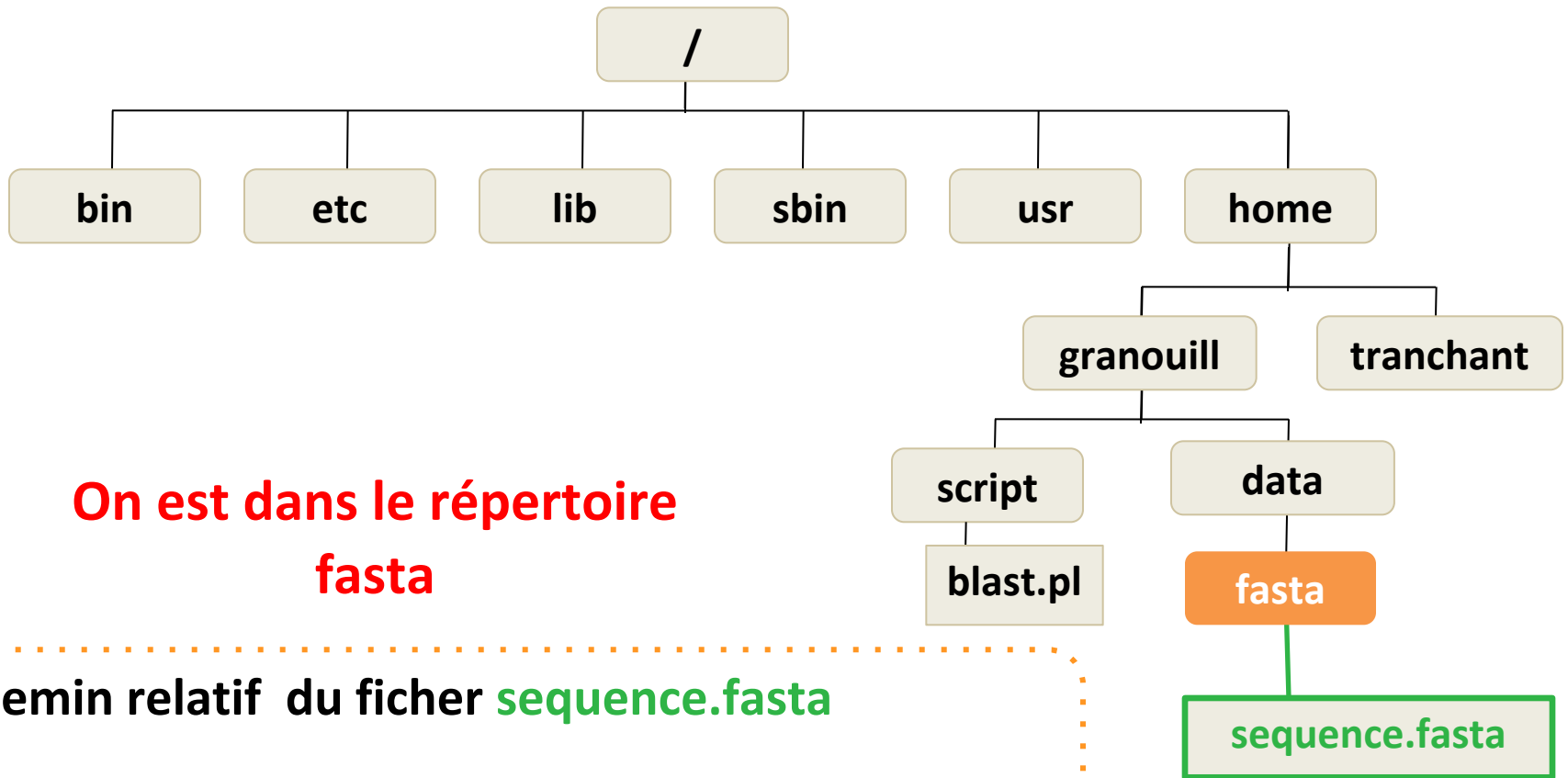
Chemin relatif du fichier **sequence.fasta**

fasta/sequence.fasta

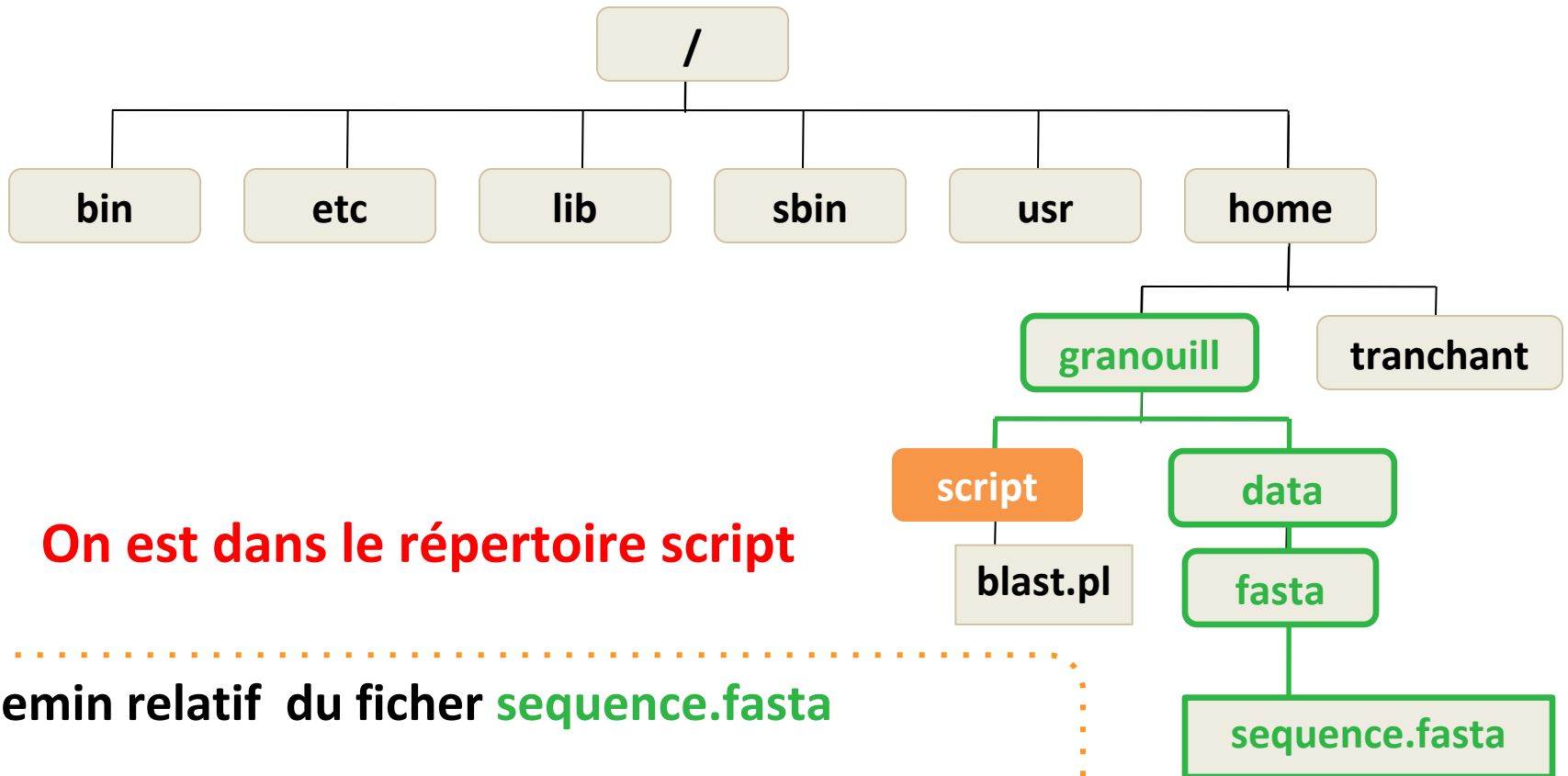
- Défini par rapport où on est dans l'arborescence
- **Ne commence jamais par /**



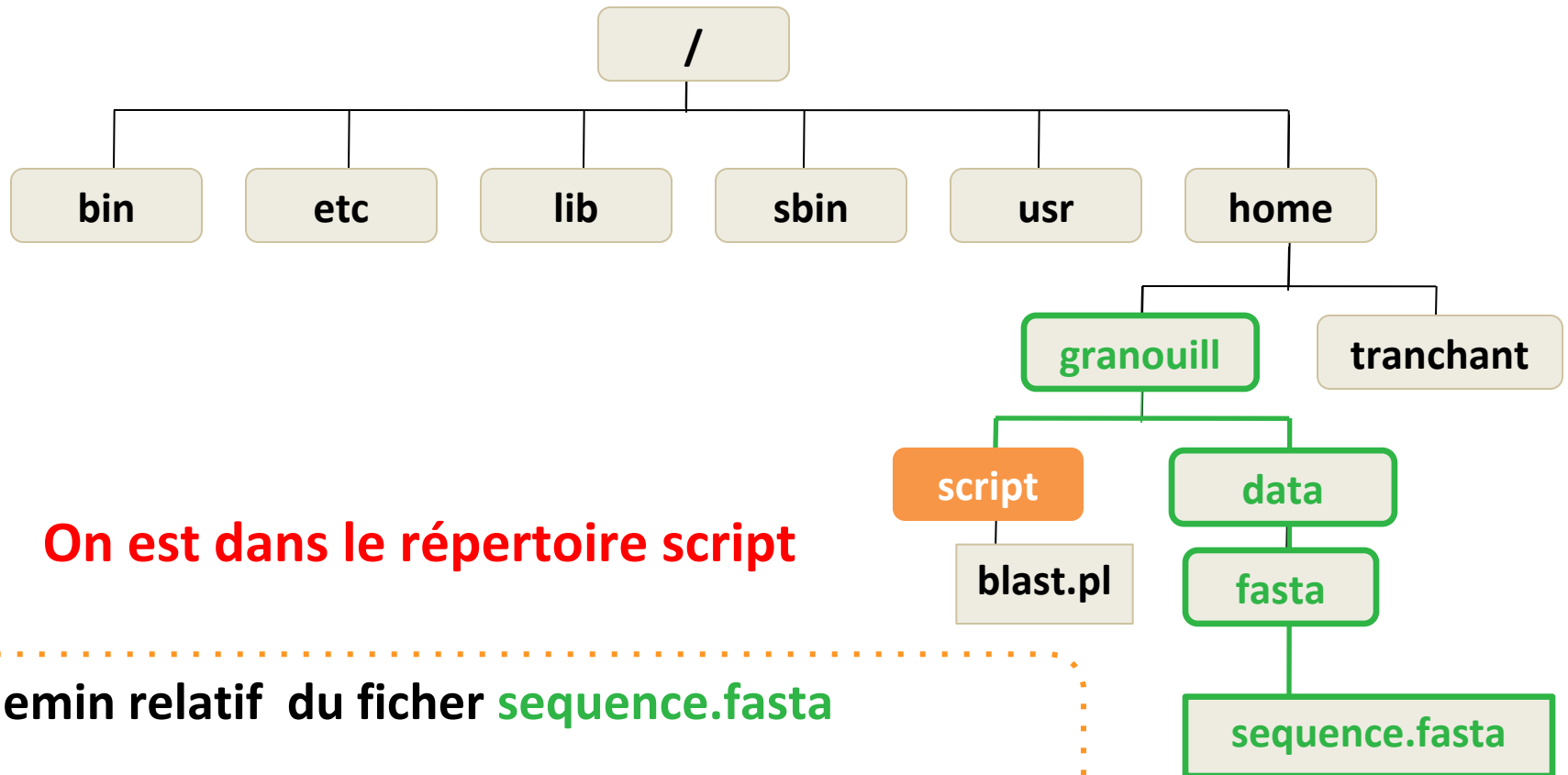
- Défini par rapport où on est dans l'arborescence
- **Ne commence jamais par /**



- Défini par rapport où on est dans l'arborescence
- **Ne commence jamais par /**



- Défini par rapport où on est dans l'arborescence
- **Ne commence jamais par /**



On est dans le répertoire script

Chemin relatif du fichier **sequence.fasta**

../data/fasta/sequence.fasta



Practice

Is

4

Go to [Practice 4](#) on our github

Interagir avec les processus

<Ctrl> + C Arrêter le processus en cours sous le terminal

<Ctrl> + Z

Tab completion

<Tab> Complète automatiquement le nom d'un fichier/
répertoire qui est en cours de saisie (fichier unique)

<Tab><Tab> Affiche la liste des différentes possibilités si le
choix n'est pas unique

Intéragir avec l'historique de commandes

Flèche bas/haut - Afficher la commande précédente/suivante
- Presser plusieurs fois pour naviguer dans
l'historique

<Ctrl> + R - Afficher la dernière commande qui contient
les caractères saisis.
- Presser les touches et commencer à taper la
commande recherchée

- Linux est sensible à la case

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

- Utiliser uniquement des lettres, chiffres et caractères _ . -
- PAS d'espaces, accents et caractères spéciaux

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

- Le suffixe des noms de fichiers (.txt, .fasta, .fa, .fq etc.) est optionnel



Travailler avec les “jokers”

Métacaractères : *, []

C'est un caractère qui peut être utilisé comme substitut de plusieurs caractères ou classes de caractère dans une recherche

Peut être utilisé avec de nombreuses commandes linux

- * N’importe quel caractère ou chaîne de caractères



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

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

KYVF.sam
KYVF.bam

ZO16.fastq
ZO16.bam

- * N’importe quel caractère ou chaîne de caractère



KYVF-01.R1.fastq

KYVF-02.R1.fastq

KYVF.sam

ZO16.fastq

KYVF-01.R2.fastq

KYVF-02.R2.fastq

KYVF.bam

ZO16.bam

```
ls *fastq
```

* N'importe quel caractère ou chaîne de caractère



KYVF-01.R1.fastq	KYVF-02.R1.fastq	KYVF.sam	ZO16.fastq
KYVF-01.R2.fastq	KYVF-02.R2.fastq	KYVF.bam	ZO16.bam

`ls *fastq`

KYVF-01.R1.fastq	KYVF-02.R1.fastq	ZO16.fastq
KYVF-01.R2.fastq	KYVF-02.R2.fastq	

- * N’importe quel caractère ou chaîne de caractère



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

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

KYVF.sam
KYVF.bam

ZO16.fastq
ZO16.bam

```
ls KYVF*fastq
```

- * N’importe quel caractère ou chaîne de caractère



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

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

KYVF.sam
KYVF.bam

ZO16.fastq
ZO16.bam

```
ls KYVF*fastq
```

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

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

* N’importe quel caractère ou chaîne de caractères

[] Tous les caractères entre crochets



KYVF-01.R1.fastq

KYVF-02.R1.fastq

KYVF.sam

ZO16.fastq

KYVF-01.R2.fastq

KYVF-02.R2.fastq

KYVF.bam

ZO16.bam

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

* N’importe quel caractère ou chaîne de caractère

[] Tous les caractères entre crochets



KYVF-01.R1.fastq

KYVF-02.R1.fastq

KYVF.sam

ZO16.fastq

KYVF-01.R2.fastq

KYVF-02.R2.fastq

KYVF.bam

ZO16.bam

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

```
= ls *. [!f]*
```

```
KYVF.sam ZO16.bam
```

```
KYVF.bam
```



Practice

ls, *

5

Go to [Practice 5](#) on our github



Commandes relative à l'arborescence de fichiers

commande `cd`

cd

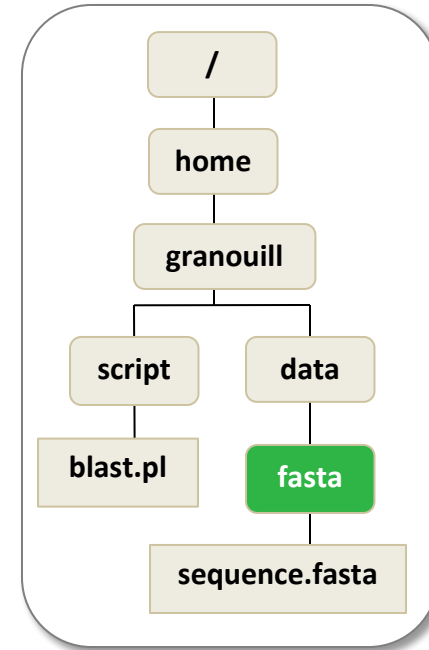
pour se déplacer dans l'arborescence

Change Directory

cd nom_repertoire (chemin absolu ou relatif)

`cd nom_repertoire(chemin absolu ou relatif)`

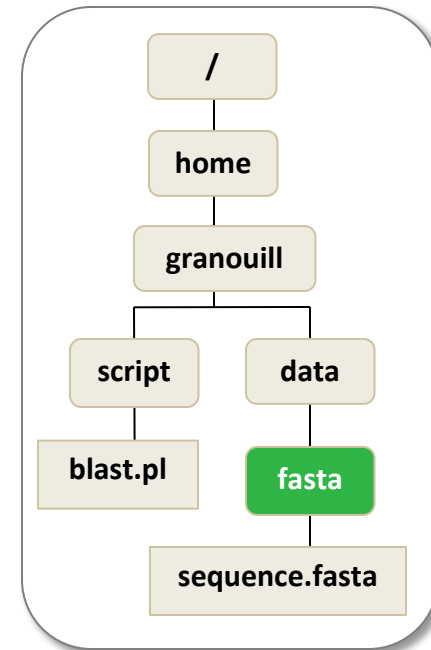
Chemin absolu :



`cd nom_repertoire(chemin absolu ou relatif)`

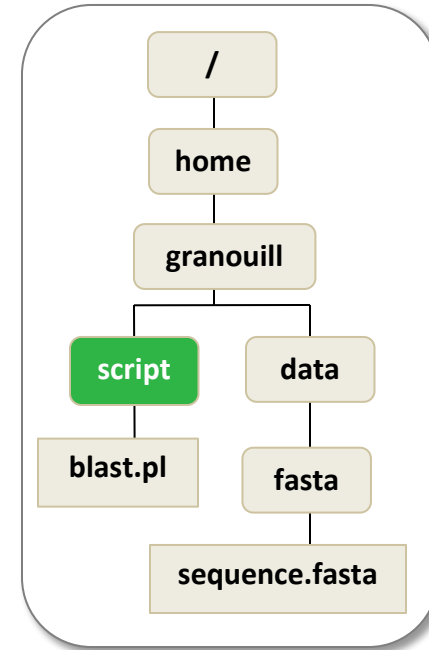
Chemin absolu :

`cd /home/granouill/data/fasta`



`cd nom_repertoire(chemin absolu ou relatif)`

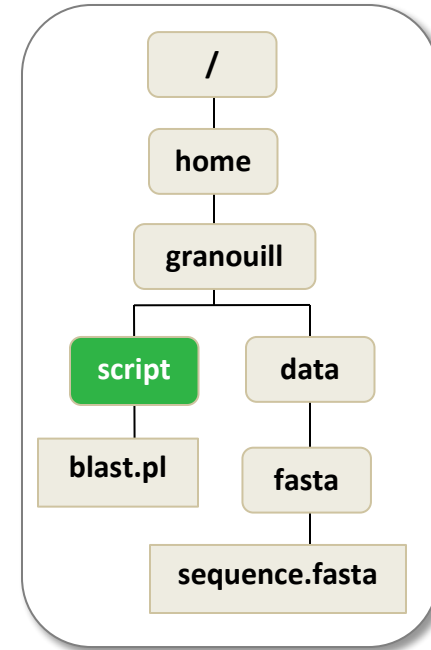
Chemin absolu :



`cd nom_repertoire(chemin absolu ou relatif)`

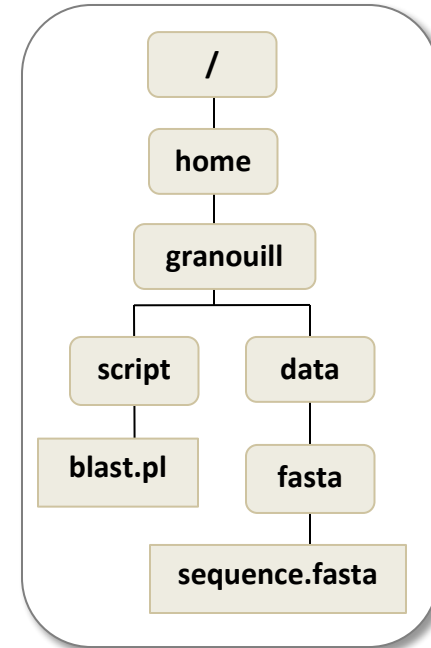
Chemin absolu :

`cd /home/granouill/script/`



`cd nom_repertoire(chemin absolu ou relatif)`

Chemin relatif :



Commande	Se déplace dans
----------	-----------------

cd directory_name	directory_name
-------------------	----------------

cd	<i>home directory</i>
----	-----------------------

cd ..	le répertoire parent
-------	----------------------

cd ../..	?
----------	---

Retour au répertoire de l'utilisateur courant home

Remonte d'1 répertoire

Remonte de 2 répertoires

Arborescence linux

pwd Affiche le chemin absolu
ls Liste tous les fichiers/répertoires
cd rep_name Se déplace dans rep_name

mkdir rep_name Crée un répertoire
rm nom_fichier Supprime un fichier
cp fichier_source repertoire_cible Copie un fichier dans un répertoire
cp fichier_source nouveau_fichier Copie un fichier sous un nouveau nom



Practice

`pwd, cd, ls`

6

Go to [Practice 6](#) on our github



Practice

`mkdir, mv, ls, cp, rm`

7

Go to [Practice 7](#) on our github



Des commandes pour éditer des fichiers et les manipuler

cat nom_fichier

Affiche le contenu d'un fichier sans pagination

```
MacBook-Pro-de-Christine:Data tranchan$ cat Data/Fasta/sequence.fasta  
>Gxbjbsjxbjs  
CCACCCCTCTTACAGTCTTCACCAAATGTCCTTTAAACTCCACCTAAAGTATCCAAAGA  
CTCGAGAAATGCTGTGCCACAACCAGCTTTTGAGTCATCCATGACCGTTGATCTTCCTTT  
GCCCCAGAGTGGGGCCTAGCACCATCTAGCTACTACTTGCCCTTCATACCCATCATTGG  
GATACCCTGAATACCTATCTTATAAGTTCCATATGGCTTATATTTCTAAGTAAGAGATGC  
ACTTAGTAAGTGCATGTCGCTTGACTTGTTTATACTCTAATGTATGATATTTATATCCC  
TATAATATAGTGTTACTAATATATGTTTGGTATTGTGTAGACTCCATTGTACCATGGTGT  
GCTAATTAGAAATAACATGCCAGCTTTGCTATTGTGGTTTGCAAGTAAAGTAAAAAAAAA  
MacBook-Pro-de-Christine:Data tranchan$
```


head Affiche les premières lignes d'un fichier
(n=10, 10 lignes par défaut) `head -n 20 script.pl`

tail affiche les dernières lignes d'un fichier
(n=10 par défaut) `tail -n 5 script.pl`

wc Compter le nombre de lignes, mots ou
caractères d'un fichier `wc script.pl`
`wc -l script.pl`



Practice

ls, head, tail, wc

8

Go to [Practice 8](#) on our github

grep

pour rechercher un motif dans une ligne

```
grep [options] motif [file1, ... ]
```

**Pour rechercher un motif, le mettre entre simple/double quote
=> ex : 'gene' ou « gene »**

grep

pour rechercher un motif dans une ligne

```
grep [options] motif [file1, ... ]
```

Option	Description
-i	Recherche le motif sans tenir compte de la casse
-c	Compte le nombre de lignes dans lesquelles le motif a été trouvées
-v	Affiche seulement les lignes sans le motif
-l	Affiche uniquement les noms de fichiers dans lesquels le motif a été trouvé



Practice

wget, grep

9

Go to [Practice 9](#) on our github

cut

Sélection de colonnes/champs d'un fichier

```
cut -d separateurColonne -f fieldNumber fileName
```

```
cut -d ":" -f1,5 /etc/passwd
```

Sélection des champs 1 et 5 dans un fichier tabulé
dont les colonnes sont séparées par un :

sort

Trier une ou plusieurs colonnes d'un fichier tabulé

```
sort -k2 fileName
```

sur la colonne 2 (tri alphanumérique)

```
sort -k2r fileName
```

sur la colonne 2, décroissant

```
sort -k2g -k1r fileName
```

sur la colonne 2 et 1

```
sort -t: -k3g fileName
```

sur la colonne 3 (nombre), le délimiteur de
colonne est :



Practice

makeblastdb, blastn
head, tail, wc, sort, cut

10

Go to [Practice 10](#) on our github



Les entrées / sorties

**pour sauvegarder la sortie d'une
commande dans un fichier**

La sortie d'une commande peut être écrite

dans un fichier avec les caractères :

>

>>

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

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

Redirection

Action

Command **>** file

- si le fichier n'existe pas : il sera créé
- si le fichier existe : efface le contenu

Command **>>** file

- si le fichier n'existe pas : il sera créé
- si le fichier existe : écrit à la fin du fichier



Practice

2

Go to [Practice 11](#) on our github

- La sortie d'une 1ère commande peut être envoyée comme input d'une 2ème commande
- Pour connecter/combinaison plusieurs commandes ensemble dans la même ligne de commande (sans fichier intermédiaire)
- Pipelines= *workflow*

cmd1 | cmd2 | cmd3

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

```
Root
```

```
troot
```

```
iroot
```

```
ctroot
```

```
//
```

```
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
```

```
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 -n 2
```



Practice

3

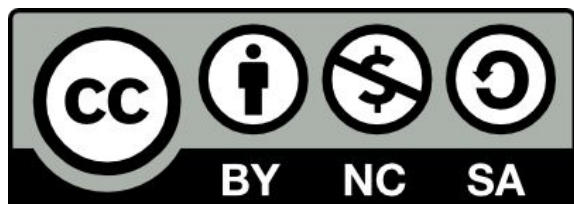
Go to [Practice 12](#) on our github

Formateurs itrop / South Green

- Christine Tranchant-Dubreuil
- Sebastien Ravel
- Alexis Dereeper
- Ndomassi Tando
- François Sabot
- Gautier Sarah
- Bruno Granouillac



Merci !



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/>



Autres commandes utiles

Compresser des fichiers **tar,gzip**

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

Décompresser une archive **gunzip, tar**

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

Afficher le contenu d'une archive **zcat**

```
zmore data.txt.gz
```

Rechercher une expression/motif dans une archive **zgrep**

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

rename

Exemple

```
rename 's/.txt/.fasta/' *
```

Description

rename the extension of all files

```
rename 'y/a-z/A-Z/' *
```

rename files in uppercase

Commande ls -l

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

Permissions

Proprio

Groupe

Taille

Heure et date de la dernière modification

Type

-Interprétation/Légendes des permissions

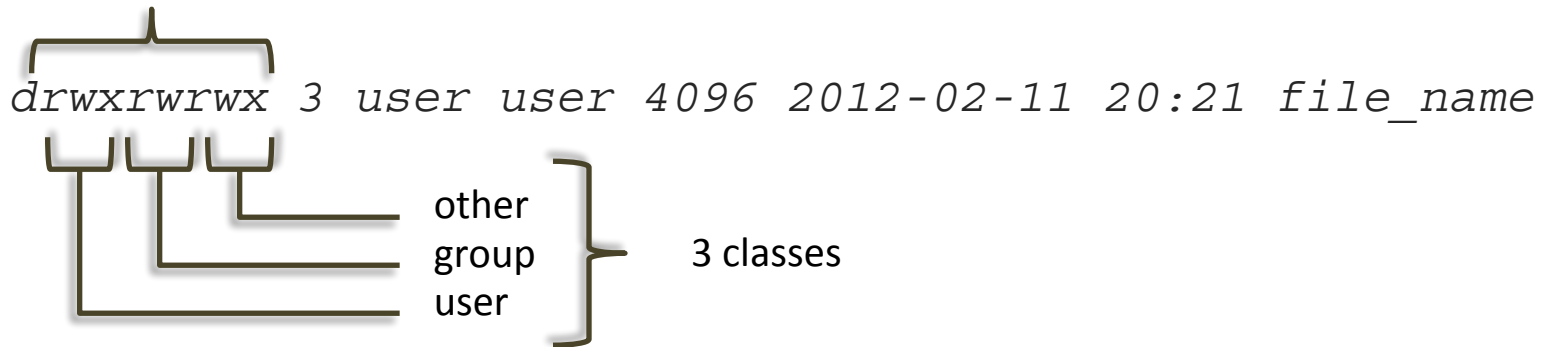
- : fichier normal

d : répertoire

l : lien symbolique

ls -l command

Permissions



3 types of permissions :

Permission	File	Directory
Read r	Ouverture et Lecture	Lister et copier les fichiers
Write w	Modiier et supprimer	Manipuler le contenu : copier, créer, modifier, écraser
Execution x	Executer le fichier	Accès seulement au fichier pour l'exécuter

commande pour la gestion des permissions : **chmod**

```
chmod <perm> file_name
```

Chaque permission = 1 valeur

R	4
W	2
X	1
none	0

Exemple

```
chmod 740 script.sh
```

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

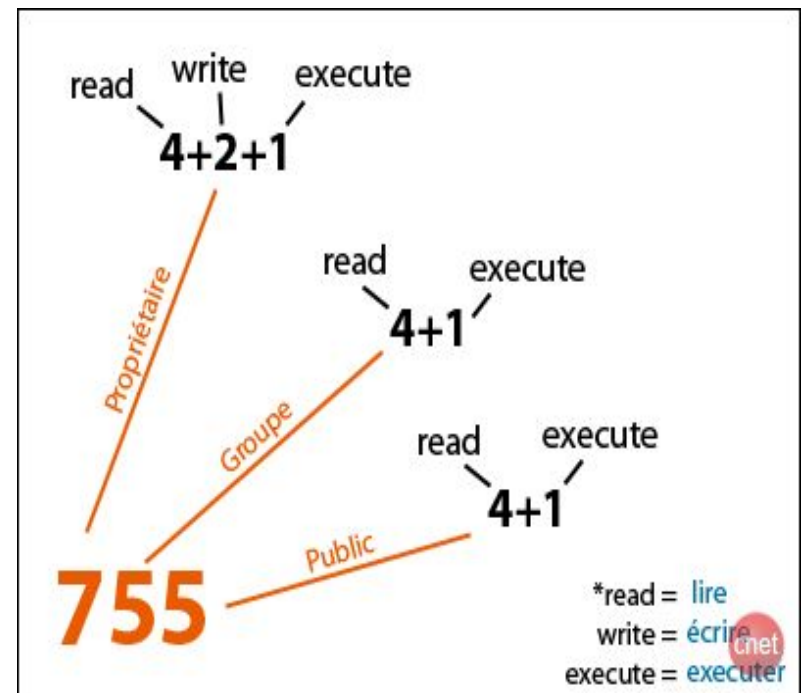
```
chmod 755 script.sh
```

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

chmod, ls

Donner le nom du propriétaire, du groupe et les droits des fichiers contenus dans le répertoire “~/Data/454-projet1/raw”

Changer les droits du fichier Scripts/blast.pl pour qu’il soit :
en lecture/écriture pour le groupe,
lecture/écriture/exécution pour le propriétaire
et lecture au public





Performing basic text transformations on a file

Sed command

```
sed "s/Search/Replace/" file
```

Replace the **SEARCH** motif by the **REPLACE** one in **FILE**

Some examples

Example

```
sed "s/ATTAAT/NN/" file
```

Description

Replace the FIRST occurrence of *ATTAAT* by *NN* in FILE

```
sed "s/ATTAAT/NN/g" file  
file
```

Replace ALL occurrences of *ATTAAT* by *NN* in FILE

```
sed "s/attaat/NN/ig" file
```

Replace ALL occurrences of *ATTAAT* by *NN* in FILE, case insensitive



Practice

6

Go to [Practice 6](#) on our github



Processing or analyzing text file

awk

awk: manipulating a file line by line...

- AWK stand for "Aho, Weinberger, Kernighan", the 3 authors
- Programming language to easily manipulate tabular files (BLAST, SAM, VCF...)
- Motif search, operation, actions...

awk: manipulating a file line by line...

- For *awk*, the input file is tabulate
- *awk* can perform computing on number and text chains
- *awk* can generate reporting following these manipulations

Syntax : awk [-F] 'program' file

Option	Description
-F	To provide the type of field delimitator (tab in standard)

Syntax : awk [-F] 'program' file

Option	Description
-F	To provide the type of field delimitator (tab in standard)

Predefined variables used by awk

Variable	Description
\$0	The whole line
NR	Number of the read line
NF	Umber of the fields in the line

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

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 every line

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
```

Print the *line number*, then the **FIRST** and **SECOND** fields

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 »; }' contact.txt
  
```

```

Helene 56
Jean 32
Julie 22
Michel 24
Richard 25
5 lines read
  
```

Perform the previous
 command then **print the
 number of lines read**

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 « Age sum : », sum; }' contact.txt
  
```

```

Helene edu
jean ri
julie adm
michel inf
richard inf
Age sum : 159
  
```

print for each line the 1st and third fields, then sum the 2nd and output the sum at the end

```
awk '$2>24 && $2<50{ print $1, " :", $2; }'  
contact.txt
```

```
Helene : 56  
jean : 32  
richard : 25
```

Print only if higher than 20 AND lower than 50

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

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



Practice

7

Go to [Practice 7](#) on our github