



HMSN118

Introduction à la bioinformatique : de l'individu à la population

#1 – Introduction à l'environnement Unix/Linux

Anna-Sophie Fiston-Lavier, PhD

Système d'exploitation (1)

Définition = un ensemble de programmes qui assurent la liaison entre tous types de plateforme informatique matériel et les applications



Systeme d'exploitation (2)

Le SE ou OS («Operating System») gère:

- Les Entrées/Sorties (E/S) ou Input/Output (I/O)
- Les fichiers
- La mémoire
- Les tâches
- Les utilisateurs
- ...

Systeme d'exploitation (3)



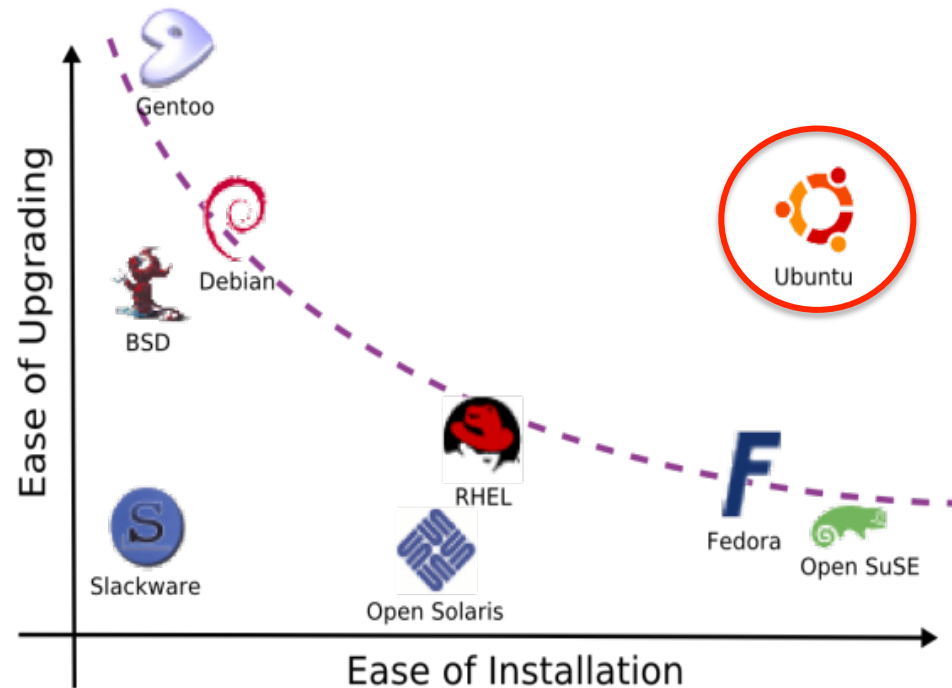
GNU is Not Unix/Linux

acronyme récursif ;)

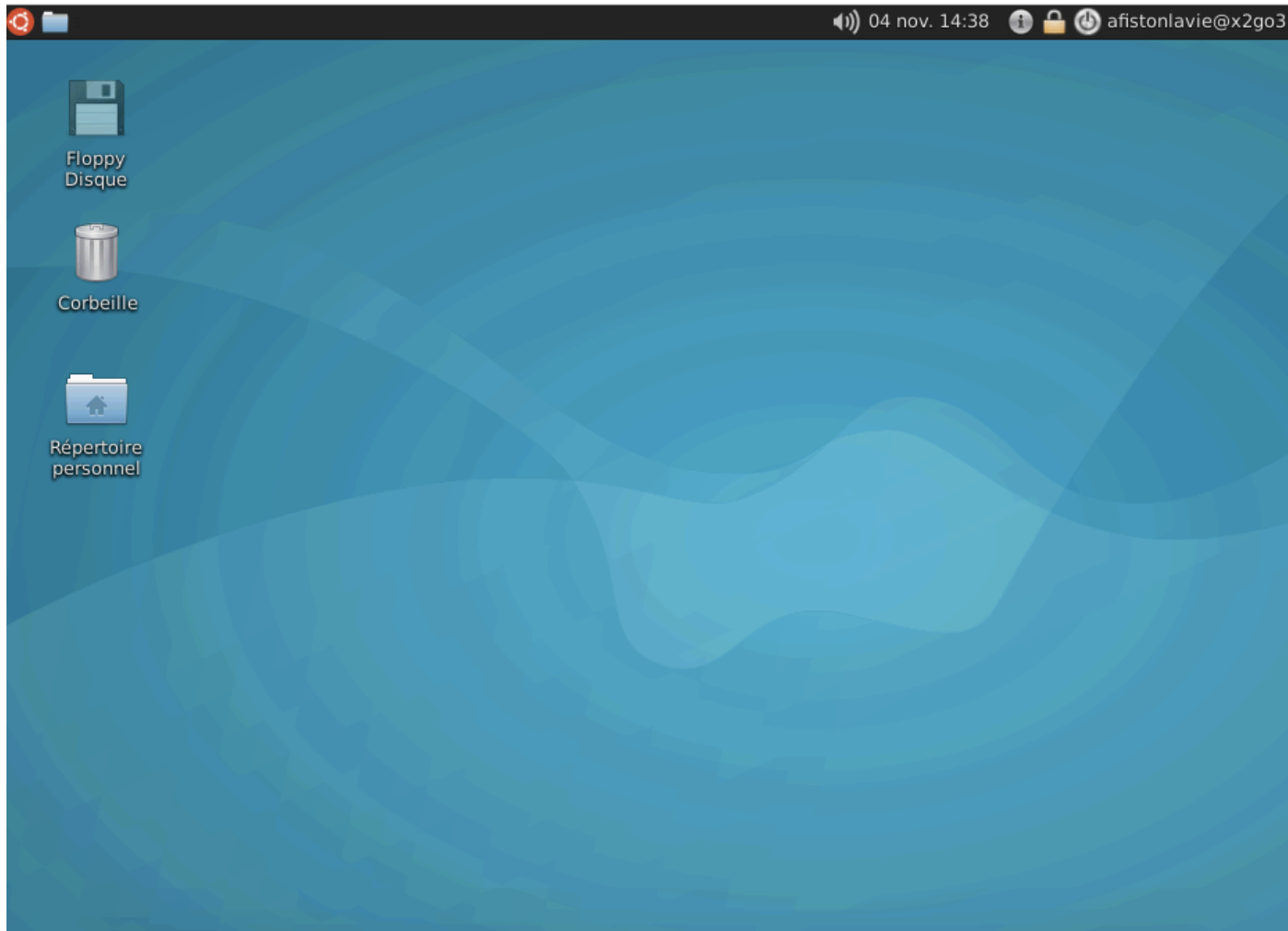


- UNIX est le SE original (1969)
- Le projet GNU démarré en 1984 par Richard Stallman a eu la mission de développer un système Unix complètement libre: LINUX
- LINUX est un SE de type généraliste, multitâche, multi-utilisateurs, partage de ressources, gestion en réseau
- Différences des sources du noyau
- Différentes d'utilisations (développement de logiciels (LINUX)/ Server internet (UNIX))

GNU/Linux distributions

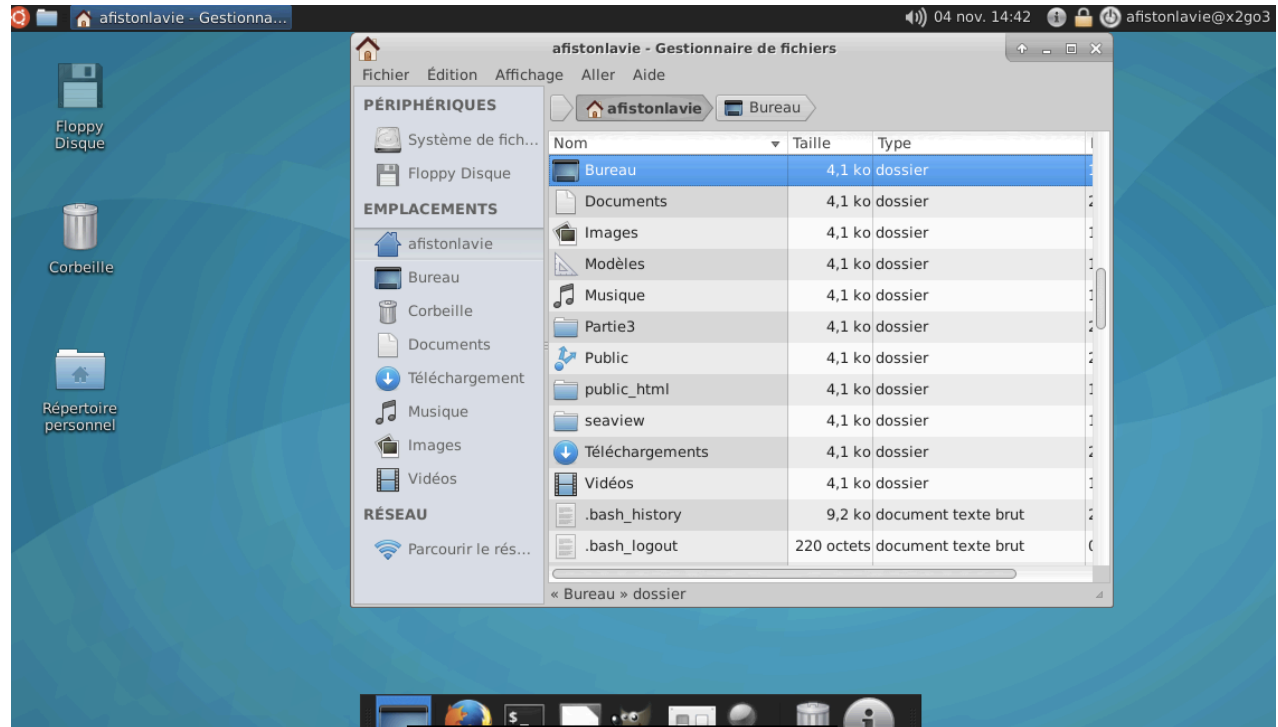


Ubuntu de la FdS

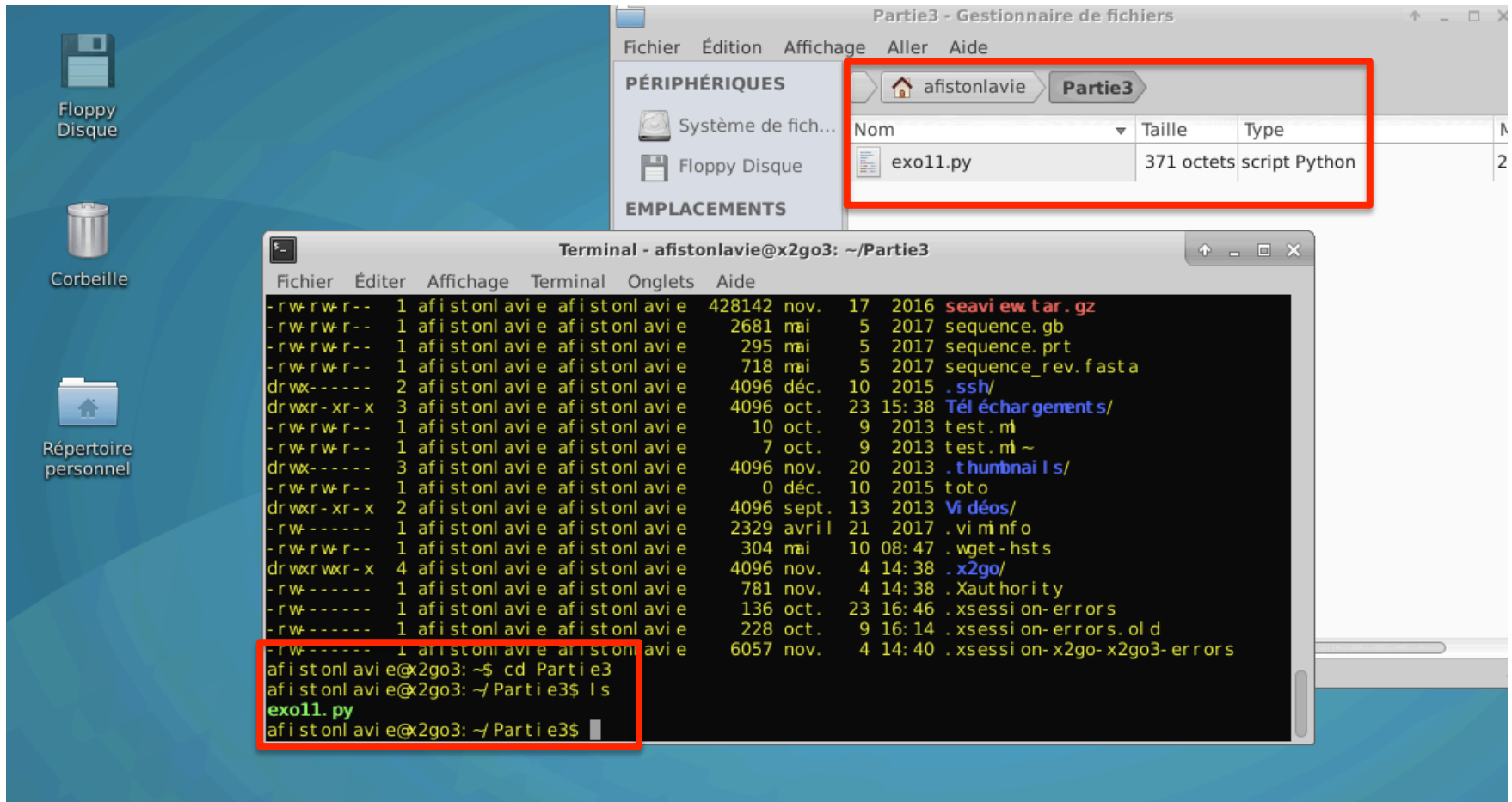


Une interface graphique facile d'utilisation

- Accès aux logiciels *via* des menus, icônes...
- Curseurs
- Souris



Un accès en ligne de commande

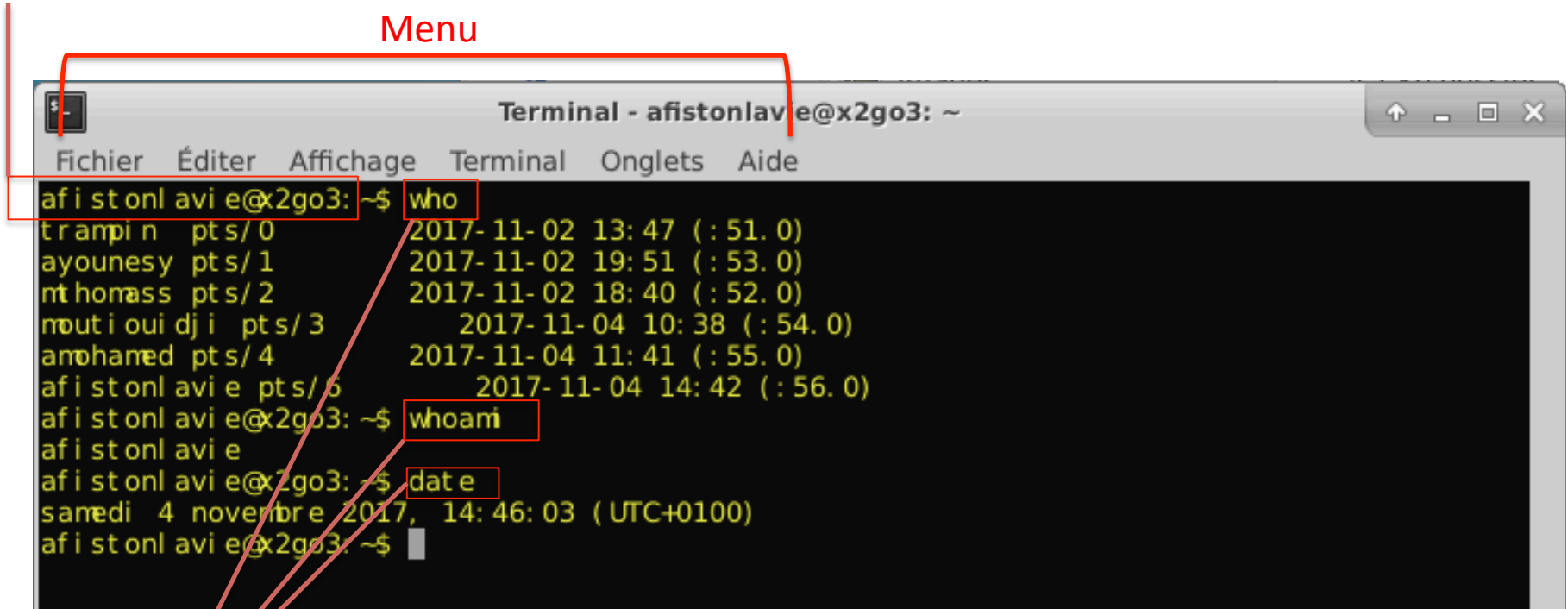


via la console (ou terminal) qui permet un affichage rudimentaire permettant d'interagir avec l'ordinateur

Le terminal

Invite ou Prompt ou zone de saisie <login>@<nom machine>:<chemin>\$

Menu



The screenshot shows a terminal window titled "Terminal - afistonlavie@x2go3: ~". The menu bar includes "Fichier", "Éditer", "Affichage", "Terminal", "Onglets", and "Aide". The prompt "afistonlavie@x2go3: ~\$" is highlighted with a red box. A red line connects this prompt to the text "Invite ou Prompt ou zone de saisie" above the slide. The terminal displays the output of the "who" command, listing users and their login times. The "whoami" command is entered and highlighted with a red box. The "date" command is also entered and highlighted with a red box. Red lines connect these command boxes to the text "Commandes" at the bottom of the slide.

```
afistonlavie@x2go3: ~$ who
trapi n pts/0      2017-11-02 13:47 (:51.0)
ayounesy pts/1      2017-11-02 19:51 (:53.0)
nthomass pts/2      2017-11-02 18:40 (:52.0)
noutiouidji pts/3    2017-11-04 10:38 (:54.0)
amohamed pts/4      2017-11-04 11:41 (:55.0)
afistonlavie pts/5    2017-11-04 14:42 (:56.0)
afistonlavie@x2go3: ~$ whoami
afistonlavie
afistonlavie@x2go3: ~$ date
samedi 4 novembre 2017, 14:46:03 (UTC+0100)
afistonlavie@x2go3: ~$
```

Commandes

Ligne de commande

Une commande est une suite de mots/symboles avec une syntaxe prédéfini qui correspondant à une ou plusieurs action(s)/opération(s) demandée(s) au SE.

Les lignes de commande sont interprétés par le **Shell** (**plusieurs types de Shell coexistent**):

1. Lecture,
2. Vérification de la syntaxe,
3. Exécution

La syntaxe d'une ligne de commande

Une commande se compose de:

- Un nom de commande
- Des options (précédées par un tiret)
- Des paramètres ou arguments (informations type nom de fichier...)

```
prompt$ commande_[-options...][-arguments...] 
```

Par exemple, la commande **ls** permet de **lister les fichiers et répertoires** présent dans le répertoire courant si non précisé

<prompt>\$: ls -l ../Assembly

Nom Options, pour afficher
les informations des
documents

Paramètres, chemin des
documents à lister

Exemple avec la commande ls

```
afistonlavi e@x2go3: ~/Partie3$ ls
ex011.py
afistonlavi e@x2go3: ~/Partie3$ ls -al
total 12
drwxrwxr-x  2 afistonlavi e afistonlavi e 4096 avril 21  2017 .
drwxrwx--x 41 afistonlavi e afistonlavi e 4096 nov.   4 14:38 ..
-rwxrwxr-x  1 afistonlavi e afistonlavi e  371 avril 20  2017 ex011.py
afistonlavi e@x2go3: ~/Partie3$ ls -l
total 4
-rwxrwxr-x 1 afistonlavi e afistonlavi e 371 avril 20  2017 ex011.py
afistonlavi e@x2go3: ~/Partie3$ ls -l ../bureau
ls: impossible d'accéder à '../bureau': Aucun fichier ou dossier de ce type
afistonlavi e@x2go3: ~/Partie3$ ls -l ../Bureau
total 0
afistonlavi e@x2go3: ~/Partie3$ ls -l ../Assembly
total 1864
-rw-rw-rw- 1 afistonlavi e afistonlavi e 811052 oct.   3 21:09 ebola1.fq
-rw-rw-rw- 1 afistonlavi e afistonlavi e 811052 oct.   3 21:09 ebola2.fq
-rw-rw-r-- 1 afistonlavi e afistonlavi e 270825 oct.  23 15:37 Ebola_fastq-20171023.zip
drwxrwxr-x 2 afistonlavi e afistonlavi e  4096 oct.  23 15:39 Raw_Data
afistonlavi e@x2go3: ~/Partie3$
```

Compactage des options: -a -l = -al

Attention à la casse: Majuscule ≠ Minuscule

Options longues ou courtes: -h = --help

Attention des options peuvent s'annuler, prendre en compte l'ordre dans certains cas!

Commencer avec la documentation

Afficher le manuel avec **la commande man**
man ls

```
LS(1)                                User Commands                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default). Sort entries
  alphabetically if none of -cftuvSUX nor --sort is specified.

  Mandatory arguments to long options are mandatory for short options too.

  -a, --all
      do not ignore entries starting with .

  -A, --almost-all
      do not list implied . and ..

  --author
```

Commencer avec la documentation

Afficher le manuel avec **la commande info**
info ls

```
LS(1)                                User Commands                                LS(1)

NAME
  ls - list directory contents

SYNOPSIS
  ls [OPTION]... [FILE]...

DESCRIPTION
  List information about the FILES (the current directory by default). Sort entries
  alphabetically if none of -cftuvSUX nor --sort is specified.

  Mandatory arguments to long options are mandatory for short options too.

-a, --all
    do not ignore entries starting with .

-A, --almost-all
    do not list implied . and ..

--author
```

Commencer avec la documentation

Afficher les options avec **l'option - h ou - - help**
ls --help

```
afistoniavie@x2go3: ~/Partie3$ ls --help
Utilisation : ls [OPTION]... [FICHIER]...
Afficher des renseignements sur les FICHES (du répertoire actuel par défaut).
Trier les entrées alphabétiquement si aucune des options -cftuvSUX ou --sort
ne sont utilisées.

Les arguments obligatoires pour les options longues le sont aussi pour les
options courtes.
  -a, --all                ne pas ignorer les entrées débutant par .
  -A, --almost-all       ne pas inclure . ou .. dans la liste
  --author                avec -l, afficher l'auteur de chaque fichier
  -b, --escape            afficher les caractères non graphiques avec des
                        protections selon le style C
  --block-size=TAILLE    convertir les tailles en TAILLE avant de les
                        afficher. Par exemple, « --block-size=M » affiche
                        les tailles en unités de 1 048 576 octets ;
                        consultez le format de TAILLE ci-dessous
  -B, --ignore-backups    ne pas inclure les entrées se terminant par ~ dans
                        la liste
```


Les droits d'accès des documents

```
hinde@hinde-VirtualBox:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 108708 janv. 17 2013 /bin/ls
```

permissions	nb liens	proprietaire	groupe	taille	date	nom
-rwxr-xr-x	1	root	root	108708	janv. 17 2013	/bin/ls

- Permissions : -rwxr-xr-x
 - premier caractère = type de (fichier -, répertoire d, ...)
 - 3 blocs de 3 caractères
 - pour le propriétaire du fichier, son groupe, tout les utilisateurs
 - droits en lecture (r), écriture (w), exécution (x)

Commandes de base:

Manipulation de fichiers

Des commandes UNIX permettent :

- Le changement des droits d'accès d'un fichier :

chmod u+x fichier1

Ajout des droits d'exécution à l'utilisateur

chmod a-rw fichier1

Suppression des droits de lecture et d'écriture pour les autres

chmod 754 fichier1

changement des droits du fichier1 (7 = rwx, 5 = r-w, 4=r--)

- Le changement de propriétaires :

chown toto:titi fichier1

changement de propriétaires utilisateur:groupe

On peut le faire de manière récursive pour tous les fichiers d'un répertoire avec l'option -R

Commandes de base :

Variables et Affichage

- La commande **echo** permet d'afficher une information stockée dans une variable d'environnement
- Les variables Shell sont affectées avec le signe « = » mais manipulées en étant appelées par leur nom précédé du signe « \$ »

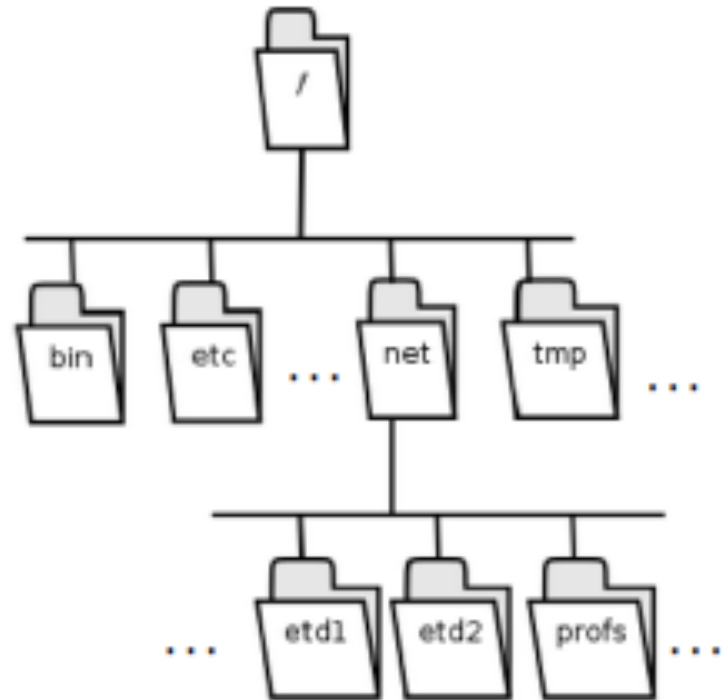
```
prompt$ code_UE=HLIN404
prompt$ Nom_UE="Intro. généraliste à l'info"
prompt$ echo "$USER_suit_ '$Nom_UE' _ (code_ $code_UE) ."
toto suit 'Intro. généraliste à l'info' (code HLIN404).
```

- Il existe des variables Shell prédéfinies: PATH, ENV, PWD...

Gestion des fichiers

Des répertoires organisés. Un répertoire = un type d'information système

- / : racine
- /tmp : fichiers temporaires
- /bin : commandes
- /etc : configuration système
- ...
- /net : fichiers partagés
- /net/edt1 : étudiants L1
- /net/profs : enseignants
- ...



Commandes de base:

Visualiser l'arborescence

- Répertoire courant (= le répertoire où je me trouve) se note « . »
- A l'ouverture de ma session, mon répertoire courant est mon HOME (/auto_home/<login> pour votre compte FdS)
- Les commandes **ls** et **tree** permettent de visualiser le contenu d'un répertoire ou chemin (= emplacement dans l'arborescence)
- L'information sur mon chemin courant peut s'obtenir *via* la commande **pwd** pour «**p**rint **w**orking **d**irectory »:

```
afistonlavi e@x2go3: ~/Partie3$ ls -l
total 4
-rwxrwxr-x 1 afistonlavi e afistonlavi e 371 avril 20 2017 ex011.py
afistonlavi e@x2go3: ~/Partie3$ tree ../Partie3
../Partie3
└── ex011.py

0 directories, 1 file
afistonlavi e@x2go3: ~/Partie3$ pwd
/auto_home/afistonlavi e/Partie3
```

Commandes de base:

Se déplacer dans l'arborescence

Un chemin peut être désigner de deux manières:

- Le chemin absolu (on part de la racine « / »)

```
afistoniavi e@x2go3: ~/Partie3$ tree /auto_home/afistoniavi e/Partie3
/ auto_home/afistoniavi e/Partie3
└─ ex011.py

0 directories, 1 file
afistoniavi e@x2go3: ~/Partie3$
```

- Le chemin relatif (on part du chemin courant)

```
afistoniavi e@x2go3: ~/Partie3$ tree ../Partie3
../Partie3
└─ ex011.py

0 directories, 1 file
```

Commandes de base:

Se déplacer dans l'arborescence

Pour se déplacer, on utilise la commande **cd** pour « **c**hange **d**irectory »:

cd ..

pour remonter d'un niveau

cd <chemin>

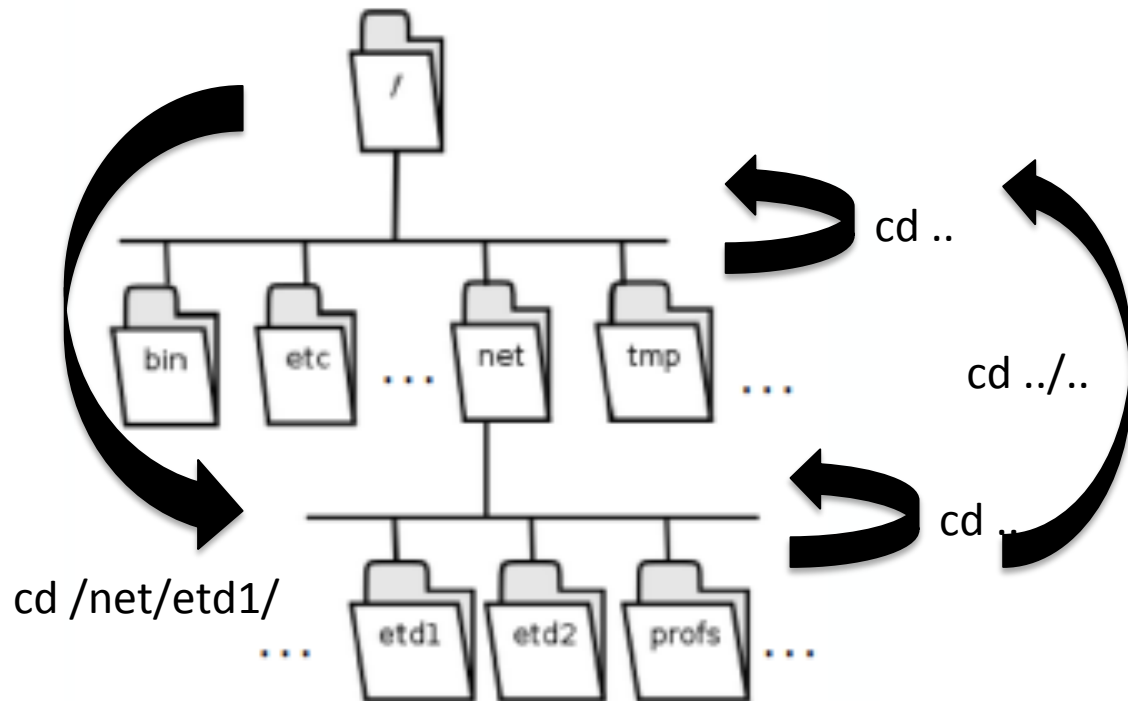
Pour aller directement dans un autre répertoire

cd -

pour revenir en arrière

cd ~ ou **cd**

Pour revenir dans votre HOME (votre répertoire)



Commandes de base:

Se déplacer dans l'arborescence

Lorsque l'on ne connaît pas (ou plus) le nom des répertoires ou que l'on veut revenir sur une commande précédente, on peut utiliser des raccourcis:

Raccourcis



complète la commande en cours



affiche une liste des différents choix de complétion



commande précédente de l'historique



commande suivante de l'historique



tue le processus en cours



recherche dans l'historique

Commandes de base:

Expression généralisée Shell

Lorsque l'on souhaite traiter plusieurs fichiers en même temps ou des fichiers présentant une syntaxe particulière, on peut utiliser les expressions:

Syntaxe

<code>?</code>	n'importe quel caractère
<code>*</code>	toute suite de caractères (y compris rien)
<code>{ abc, def, ... }</code>	ensemble de motifs possibles
<code>[1-3f-h]</code>	un des caractères parmi 1, 2, 3, f, g, h
<code>[^1-3]</code>	n'importe quel caractère sauf 1, 2 et 3

Commandes de base:

Création de répertoires et fichiers

La commande **mkdir** pour « **make directory** » permet de créer un ou plusieurs répertoire(s):

```
mkdir rep1
```

```
mkdir rep1 rep2 ... rep10
```

Il existe plusieurs procédures pour créer un fichier:

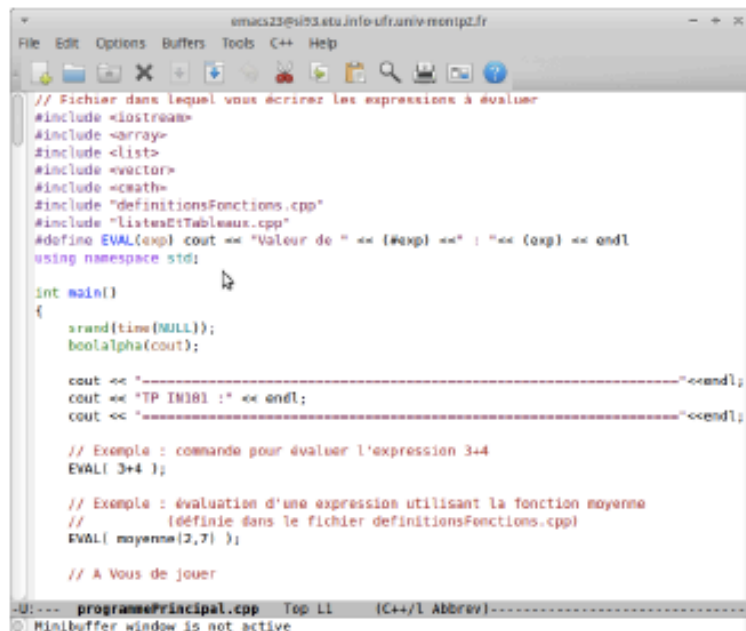
- La commande **touch**: **touch fichier1**
- La redirection : **echo "blabla" > fichier1**
 - **>** # création d'un fichier et écriture
 - **>>** # écriture à la fin d'un fichier pré-existant
 - **>&2** # redirection de l'affichage de la commande vers le canal 2
 - **<** # redirige le flux d'entrée depuis la source
- Edition d'un texte dans un éditeur type **emacs**



L'éditeur de texte Emacs

*Editing **MAC**roS running on Text Editor and Corrector*

Plusieurs éditeurs de texte existent (gedit,nano...). Je recommande d'utiliser Emacs qui présente de nombreuses fonctionnalités dédiées à la programmation



```
// Fichier dans lequel vous écrivez les expressions à évaluer
#include <iostream>
#include <array>
#include <list>
#include <vector>
#include <cmath>
#include "definitionsFonctions.cpp"
#include "ListesEtTableaux.cpp"
#define EVAL(exp) cout << "Valeur de " << {exp} << " : " << (exp) << endl
using namespace std;

int main()
{
    srand(time(NULL));
    bool alpha(cout);

    cout << "-----" << endl;
    cout << "TP IN181 :*" << endl;
    cout << "-----" << endl;

    // Exemple : commande pour évaluer l'expression 3+4
    EVAL( 3+4 );

    // Exemple : évaluation d'une expression utilisant la fonction moyenne
    // (définie dans le fichier definitionsFonctions.cpp)
    EVAL( moyenne(2,7) );

    // A Vous de jouer
}
```

Ces commandes sont accessibles à partir des menus et à partir du clavier avec les raccourcis :

Commande	Raccourci clavier
Ouvrir un fichier	<Ctrl-x><Ctrl-f>
Sauvegarder le fichier	<Ctrl-x><Ctrl-s>
Scinder la fenêtre	<Ctrl-x>2 (et 3)
N'afficher qu'une fenêtre	<Ctrl-x>1
Aller à un numéro de ligne	<Alt-g>g

Commandes de base:

Renommer et déplacer des documents

La commande **mv** pour « **move** » permet de:

- Déplacer un fichier ou répertoire:

```
mv fichier1 rep1/
```

Ici on déplace le fichier1 dans le répertoire 1

```
mv fichier2 /auto_home/<login>/rep2/
```

Ici on déplace le fichier2 dans le répertoire 2

- Renommer un fichier ou répertoire:

```
mv fichier1 fichier2
```

Ici on renomme le fichier fichier1 par le fichier fichier2

```
mv rep1 rep2
```

Ici on renomme le répertoire rep1 par le répertoire rep2

Attention au dernier "/".

Commandes de base:

Suppression de répertoires et fichiers

La commande **rmdir** pour « **re**move **di**rectory » permet de supprimer un ou plusieurs répertoire(s):

```
rmdir rep1
```

```
rmdir rep1 rep2 ... rep10
```

La commande **rm** pour « **re**move » permet de supprimer un ou plusieurs fichier(s):

```
rm rep1
```

```
rm rep1 rep2 ... rep10
```

Attention aux options du rm afin d'éviter de supprimer tous les fichiers :

```
rm *
```

Commandes de base:

lecture de fichiers

Les commandes qui permettent de lire un fichier sont:

cat fichier1 fichier2 ...

Lecture de plusieurs fichiers les uns après les autres

Possibilité de concaténer les fichiers en redirigeant vers un nouveau fichier

more fichier1

Lecture du fichier1 page par page (tapez sur la barre d'espace ou Z pour passer à la page suivante et Q pour quitter)

less fichier1

Lecture du fichier1 page par page avec la possibilité de retour en arrière (tapez sur la barre d'espace ou Z pour passer à la page suivante, touches de flèches pour se déplacer et Q pour quitter)

Commandes de base: lecture de fichiers

D'autres commandes qui permettent de lire une partie de fichier:

head fichier1

Lecture des premières lignes du fichier1 (par défaut les 5 premières)

tail fichier1

Lecture des dernières lignes du fichier1 (par défaut les 5 dernières)

Il est possible d'afficher plus de lignes (voir option -n)

Commandes de base:

Manipulation de fichiers

Des commandes UNIX permettent :

- L'analyse d'un fichier :

wc -c fichier1 #compte le nombre d'octets

wc -m fichier1 #compte le nombre de caractères

wc -w fichier1 #compte le nombre de mots

wc -l fichier1 #compte le nombre de lignes

- Le trie d'un fichier :

sort -nk 1,2 fichier1

trie le fichier dans l'ordre numérique des colonnes 1 puis 2

- La recherche d'information dans un fichier :

grep "test" fichier1

recherche du motif test dans le fichier1

Commandes de base:

Recherche d'informations dans un fichier

grep test fichier1

recherche du motif "test" dans le fichier1

grep . fichier1

recherche de n'importe quel motif dans le fichier1

grep t* fichier1

recherche du motif "t" zéro ou plusieurs fois dans le fichier1

grep t+ fichier1

recherche du motif "t" au moins une fois dans le fichier1

grep t? fichier1

recherche du motif "test" zéro ou une fois dans le fichier1

grep "^test" fichier1

recherche des lignes qui commencent par "test" dans le fichier1

grep "test\$" fichier1

recherche des lignes qui finissent par "test" dans le fichier1

grep [0-9] fichier1

recherche des lignes qui contiennent au moins un chiffre dans le fichier1

grep [a-zA-Z] fichier1

recherche des lignes qui contiennent une lettre entre a et z en majuscule ou minuscule

Commandes de base: manipuler un fichier

sed s/"test"/"TEST"/g fichier1

Recherche du motif "test" et le remplacer par le motif "TEST" dans le fichier1

tr -d "\n" fichier1

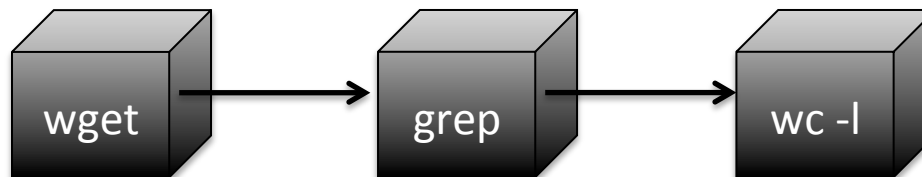
Suppression des retours chariot "\n"

cut -f 1 fichier1

Extraire la colonne 1 du fichier1 (si fichier tabulé)

Enchainement de commandes

- Le signe “|” permet d’enchaîner ou piper des commandes les unes après les autres



```
2-64-49:~ asfiston$ wget http://www.umontpellier.fr -O - -o/dev/null | grep src.*http | wc -l
20
2-64-49:~ asfiston$ wget http://www.umontpellier.fr -O - -o/dev/null > tmp1
2-64-49:~ asfiston$ grep src.*http tmp1 > tmp2
2-64-49:~ asfiston$ wc -l tmp2
20 tmp2
```

Script Shell

- On peut aussi écrire un script Shell pour enchaîner ces commandes. Pour cela, il faut:
 1. Stocker les commandes dans un fichier avec l'extension **.sh** (**test_script.sh**)
 2. Ajouter les droits d'exécution au fichier
 3. Exécuter le fichier en tapant : **./test_script.sh** afin de préciser l'emplacement du script (sinon rajouter le script à la variable **PATH**)

```
E12-64-49:~ asfiston$ cat test_script.sh
wget http://www.umontpellier.fr -O - -o/dev/null > tmp1
grep src.*http tmp1 > tmp2
wc -l tmp2
E12-64-49:~ asfiston$ chmod 777 test_script.sh
E12-64-49:~ asfiston$ ./test_script.sh
  20 tmp2
```

4. On peut rajouter en début de fichier la ligne **#!/bin/sh**

```
1  #!/bin/sh
2
3  PL_DIR="$HOME/.playlists"
4  PL_FILES="$PL_DIR/*.m3u"
5  SERVER="localhost:5555"
6
7  if [ -d $PL_DIR ]; then
8      if ls $PL_FILES > /dev/null 2>&1; then
9          if vlc -Z -I rc --rc-fake-tty --quiet --rc-host $SERVER $PL_FILES > /
dev/null 2>&1; then
10             echo -e "End_of_vlc_deamon"
11         else
12             echo -e "Error: vlc isn't running..." > /dev/stderr
13         fi
14     else
15         echo -e "Aucun_fichier_trouve_dans_le_repertoire '$PL_DIR'." > /dev/
stderr
16         exit 2
17     fi
18 else
19     echo -e "Erreur, le repertoire '$PL_DIR' n'existe pas." > /dev/stderr
20     exit 1
21 fi
22
23 exit 0
```

Gestion des processus

- Processus = activité (exécution d'un programme) + données pour la gérer
- Ne pas confondre processus et programme ou application
 - un même programme ou application peut avoir plusieurs exécutions simultanées
- Les processus sont protégés les uns des autres par le système
 - espace d'adressage de chaque processus
 - partage de ressources (processeurs, mémoire, réseau, disque, etc.)

Gestion des processus

- Un système d'exploitation multi-tâche comme Linux permet d'exécuter plusieurs processus de façon « quasi-simultanée »
- S'il y a plusieurs processeurs, l'exécution des processus est distribuée de façon équitable sur ces processeurs
 - rôle de l'ordonnanceur

Gestion des processus

- Plusieurs actions possibles
 - lancer un processus
 - afficher la liste des processus
 - tuer un processus
 - commande kill
 - communiquer entre processus
 - exemple :

```
ls /usr/share/doc | more
```

la sortie de **ls /usr/share/doc** est redirigée vers l'entrée de **more**

Gestion des processus

```
top - 16:49:25 up 12:21,  2 users,  load average: 0,11, 0,06, 0,05
Tasks: 152 total,   1 running, 151 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1,3 us,  0,7 sy,  0,0 ni, 98,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0
KiB Mem:  4136176 total,  987012 used,  3149164 free,   85512 buffers
KiB Swap: 4192252 total,    0 used,  4192252 free,  467124 cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1839	hinde	20	0	413m	174m	35m	S	1,3	4,3	2:29.47	compiz
1001	root	20	0	176m	48m	13m	S	0,7	1,2	0:51.83	Xorg
3912	hinde	20	0	5228	1332	984	R	0,3	0,0	0:00.63	top
1	root	20	0	3916	2272	1368	S	0,0	0,1	0:00.79	init
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0,0	0,0	0:00.36	ksoftirqd/0
5	root	0	-20	0	0	0	S	0,0	0,0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0,0	0,0	0:04.63	kworker/u2:0
7	root	rt	0	0	0	0	S	0,0	0,0	0:00.00	migration/0
8	root	20	0	0	0	0	S	0,0	0,0	0:00.00	rcu_bh
9	root	20	0	0	0	0	S	0,0	0,0	0:00.47	rcu_sched

Quizz (1/5)

- Quelle commande vous permet de savoir où vous vous êtes positionnés dans l'arborescence?

Quizz (1/5)

- Quelle commande vous permet de savoir où vous vous êtes positionnés dans l'arborescence?

`pwd`

Quizz (2/5)

- Quelle commande vous permet d'afficher le contenu d'un répertoire? Que se passe t-il si l'on rajoute les options `-a` puis `-l` ?

Quizz (2/5)

- Quelle commande vous permet d'afficher le contenu d'un répertoire? Que se passe t-il si l'on rajoute les options `-a` puis `-l` ?

`ls` #liste les documents

`ls -a` #liste les documents et fichiers cachés

`ls -l` #liste les documents avec le détail des droits

Quizz (3/5)

- Quelle commande vous permet d'aller dans le répertoire Bureau? Puis de revenir dans votre répertoire HOME?

Quizz (3/5)

- Quelle commande vous permet d'aller dans le répertoire Bureau? Puis de revenir dans votre répertoire HOME?

`cd /auto_home/<login>/Bureau`

`cd ..` Ou `cd -` ou `cd` ou `cd ~`

Quizz (4/5)

- Quelle commande vous permet de lister les fichiers image au format jpg ? Et les compter ?

Quizz (4/5)

- Quelle commande vous permet de lister les fichiers image au format jpg ? Et les compter ?

```
ls -l *.jpg ou ls -l jpg$ | wc -l
```

Quizz (5/5)

- Quelle commande vous permet créer en une seule fois un répertoire Omics et trois nouveaux répertoires : TP1, TP2 et TP3?
Renommer TP1 par TP_UNIX.

Quizz (5/5)

- Quelle commande vous permet créer en une seule fois un répertoire Omics et trois nouveaux répertoires : TP1, TP2 et TP3? Renommer TP1 par TP_UNIX.

```
mkdir Omics Omics/TP1 Omics/TP2  
Omics/TP3 | mv Omics/TP1 Omics/  
TP_UNIX
```