

Man_Shell

Mise à niveau Système d'exploitation Unix

Comment utiliser sa puissance

Antoine Cornuéjols – Christine Martin – Chloé Vigliotti

AgroParisTech – INRAe MIA Paris-Saclay

EKINOCS research group

Objectifs du cours

1. Savoir utiliser l'environnement Unix
2. Avoir eu un 1^{er} contact avec les expressions régulières
3. Notions de script bash et d'utilisation de Python pour le maniement des fichiers
4. Savoir installer un logiciel

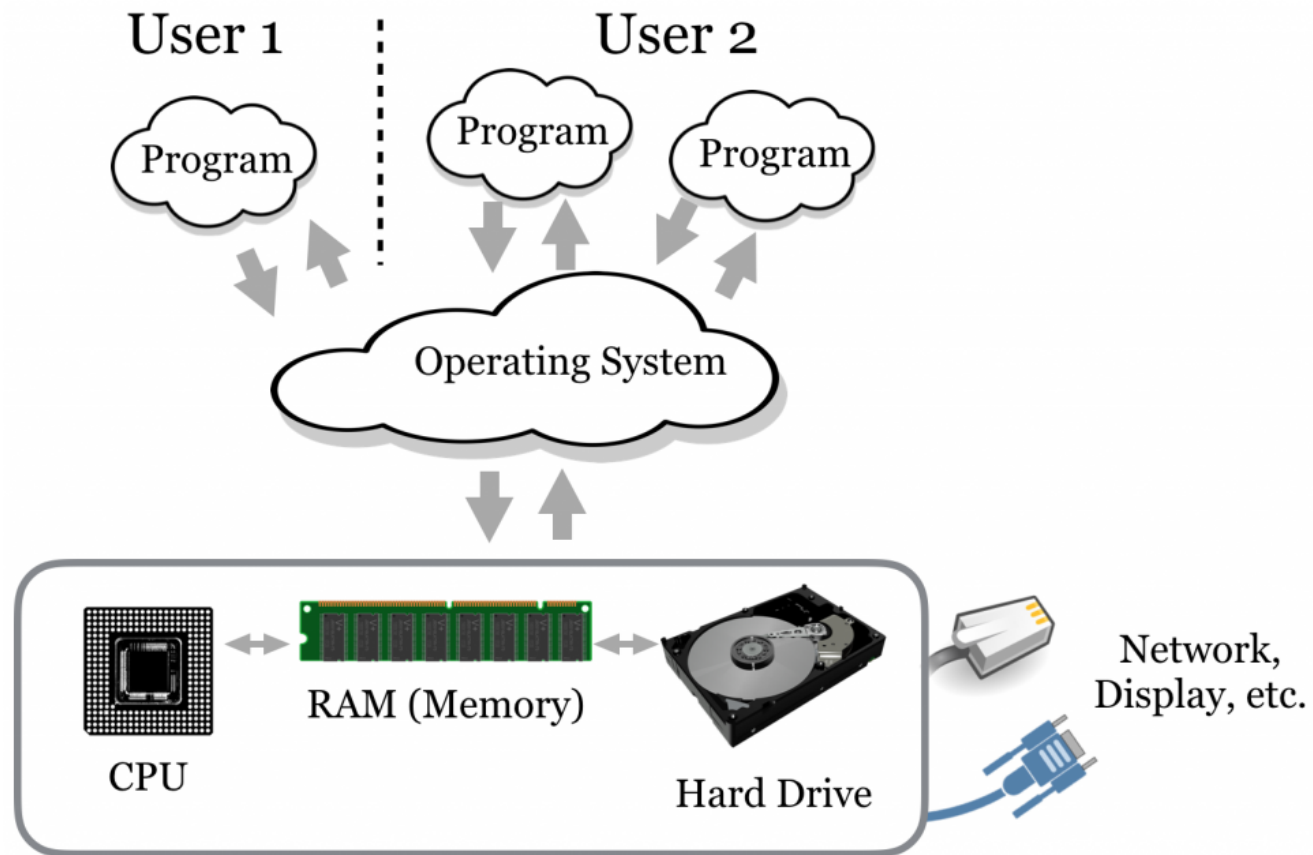
Plan

1. L'environnement Unix
2. Travailler avec des motifs : les expressions régulières
3. Scripts Bash et programmes Python
4. L'installation de logiciels

Système d'exploitation

- Qu'est-ce qu'un **système d'exploitation** (Operating System (OS))?
 - Tout le fonctionnement d'un ordinateur est **contrôlé** par un système d'exploitation (*Operating System* ou OS)
 - Exemple : lorsque plusieurs programmes tournent ensemble, c'est l'OS qui alloue les ressources entre eux et les interrompt si nécessaire.
 - **Unix** est un exemple d'un OS, très répandu (toutes les machines de calcul scientifique déjà)
 - Développé dans les années 70 (après Multics)
 - **Mac OS X** et **Ubuntu** (utilisant Linux), par exemple, sont basés sur Unix (mais Windows ne l'est pas)
 - En général, les OS ont **deux types d'interface**
 - **Graphique** (GUI : Graphical User Interface)
 - **En ligne de commande** (CLI : Command Line Interface)

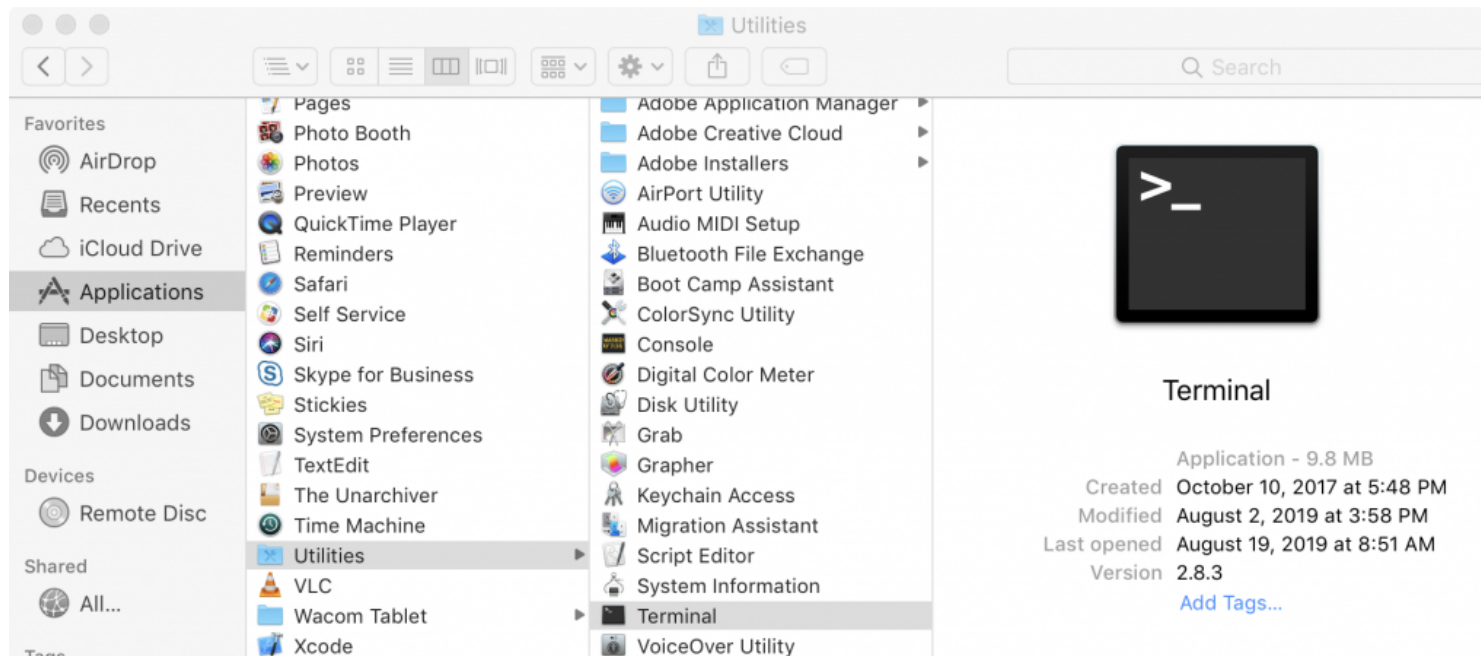
Système d'exploitation



From [Shawn T. O'Neil (2019) « A primer for Computational Biology », Oregon State University]

Accès à un système Unix

- Mac OS X ou Linux : vous avez un accès « direct »
 - Sous **Linux** : ctrl + alt + T
 - Sous **Mac OS X**
 - Icône **Terminal** dans le Launchpad
 - OU :



Accès à un système Unix sous Windows

- Accès à une **machine virtuelle** Unix ou Linux
- Utilisation de **WSL** (Windows Sous Linux)
 - Le Sous-système Windows pour Linux permet aux développeurs d'exécuter un environnement GNU/Linux (et notamment la plupart des utilitaires, applications et outils en ligne de commande) **directement sur Windows**, sans modification et tout en évitant la surcharge d'une machine virtuelle traditionnelle ou d'une configuration à double démarrage.
 - <https://learn.microsoft.com/fr-fr/windows/wsl/>

Accès à un système Unix

- Windows

- Il faut installer un client ssh (secure shell).

- Le plus communément utilisé est **putty** (<https://www.putty.org/>)
- Installer la version 32 ou 64 bits correspondant à votre machine

- Se connecter depuis PUTTY dans l'onglet session

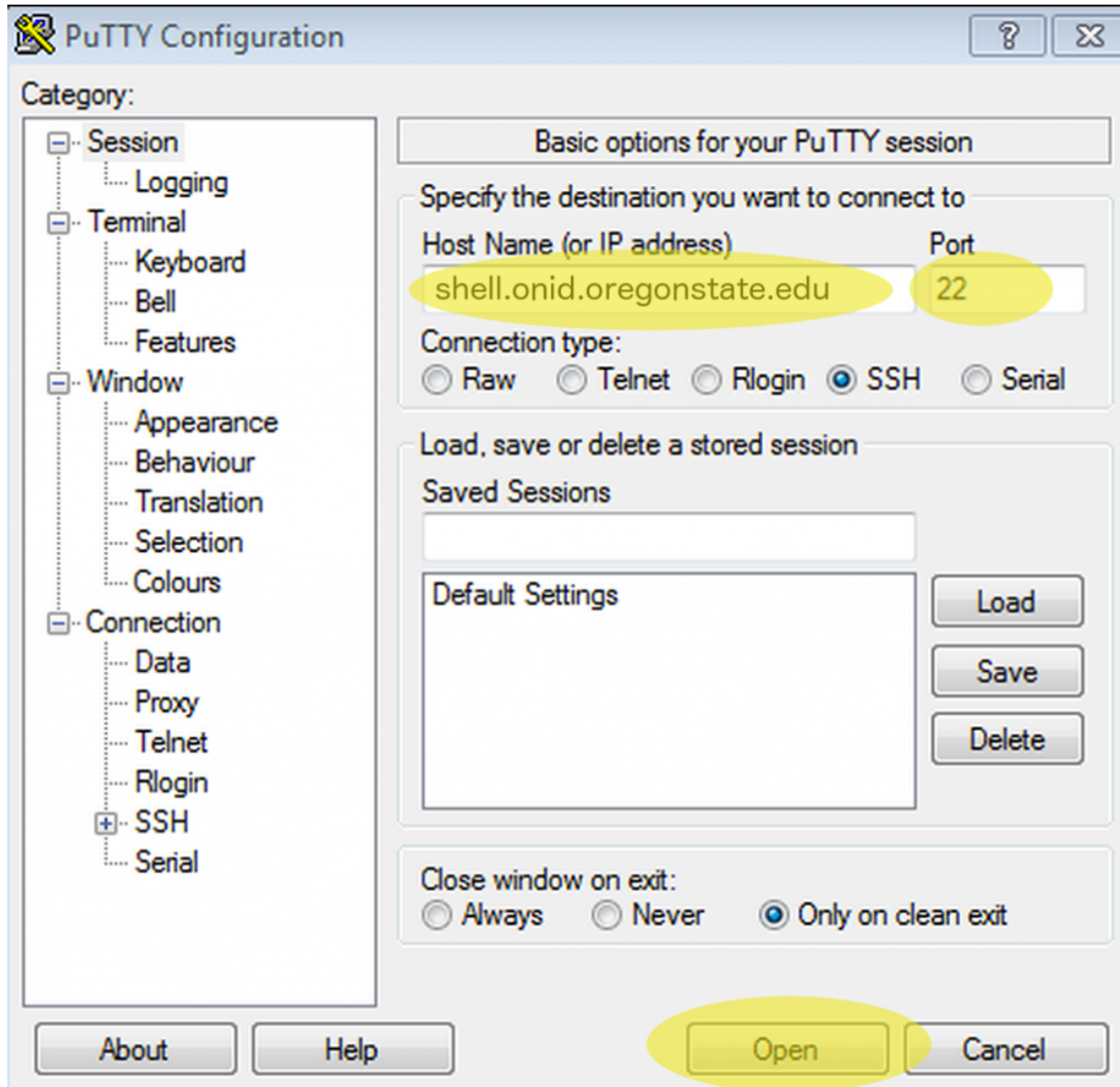
- Renseigner le champ « host name (or IP address) avec l'information
 - ✓ nomutilisateur@nomserveur
 - ✓ Puis cliquez sur « open » (en laissant le port 22)

- ssh utilise 4 informations

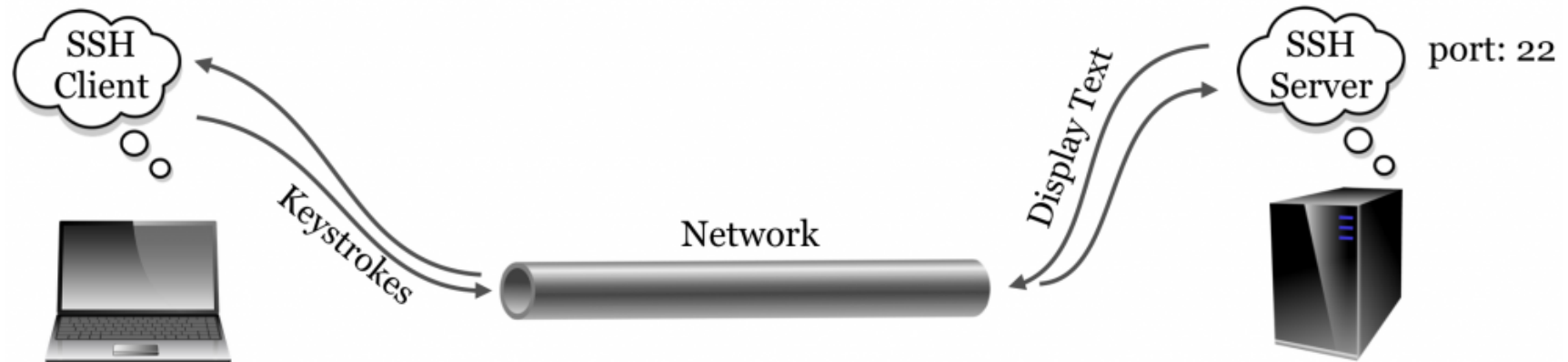
- L'adresse du serveur
- Le nom de l'utilisateur
- Le mot de passe
- Le port (en général, le port 22)

```
ssh <nom_utilisateur>@<ipaddress> -p <num_port>
```


PUTTY



ssh



- Le **serveur** ssh et le **client** ssh communiquent par le **protocole ssh**
 - **Sécurisé** (par cryptographie à clé publique avancée)
 - Le **client** envoie des commandes
 - Le **serveur** envoie du texte
 - Par convention, SSH se connecte sur le **port 22** (et HTTP sur le port 80)

Le shell

- Notion de shell
 - **Interface** permettant d'interagir avec un système d'exploitation
 - Un OS est composé d'un **noyau** (kernel) et d'une **coque** (shell)
 - le **shell** est un programme qui reçoit des commandes qu'on va écrire depuis le clavier (ou par une interface graphique + souris) et qui les passe au système d'exploitation afin qu'elles soient exécutées.

Bash

- **Shell : Interface** permettant d'interagir avec un système d'exploitation
 - nous allons devoir **envoyer nos commandes** sous un certain **format** compréhensible par notre OS. Ce « format » est en fait **un langage de programmation** spécifique au shell utilisé.
 - Le plus utilisé est **Bash** (*“Bourne Again SHell”*).
 - Bash est notamment le shell utilisé par défaut par les systèmes OS X (en fait zsh depuis Catalina)
- **Terminal**
 - Le **Terminal** est l'interface de ligne de commande de **Mac**.
 - Nous allons utiliser le Terminal pour envoyer nos commandes à notre OS et pour communiquer avec lui.
 - L'invite de commande ou “command prompt” ou CMD est l'interpréteur de ligne de commande **Windows**.

Bash

- Ouverture de **Terminal**



```
antoinecornuejols — -zsh — 80x24
Last login: Wed Aug 16 18:28:13 on ttys001
antoinecornuejols@MacBook-Pro-de-ANTOINE-5 ~ % █
```

- Lorsqu'on ouvre le Terminal, une **fenêtre** noire ou blanche s'ouvre. Cette fenêtre contient un **invite de commande**, c'est-à-dire une ligne nous indiquant que le shell attend qu'on lui passe des commandes.
- Par défaut, cette ligne contient le **nom de votre machine** suivi de **deux points** suivi du caractère **tilde** (~) suivi de votre **nom d'utilisateur** suivi du **signe dollar** (\$) (ou % pour Zsh)).
- Le **tilde** est une abréviation pour indiquer qu'on se situe actuellement **dans le dossier "home"**, c'est-à-dire dans le répertoire de base lié à notre nom d'utilisateur.
- Le signe **dollar** (ou %) indique qu'on est **connecté en tant qu'utilisateurs classiques** avec des privilèges normaux.

Les commandes Bash

et le système de fichiers

Premiers pas et variables d'environnement

```
oneils@atmosphere ~$ echo hello there  
hello there
```

```
oneils@atmosphere ~$ echo $USER  
oneils
```

```
oneils@atmosphere ~$ echo $0  
-bash
```

```
oneils@atmosphere ~$ tcsh  
172:~> echo $0  
tcsh
```

```
172:~> exit  
exit  
oneils@atmosphere ~$ echo $0  
-bash
```

...

Se diriger dans le système de fichiers

- Quand vous vous loggez pour la 1^{ère} fois, vous arrivez dans votre « home directory »
- PWD : Present Working Directory

```
oneils@atmosphere ~$ echo $HOME
/home/oneils
oneils@atmosphere ~$ echo $PWD
/home/oneils
oneils@atmosphere ~$ pwd
/home/oneils
```


Se diriger dans le système de fichiers

- Vous pouvez avoir la **liste des fichiers** de votre PWD par la commande **ls**

```
oneils@atmosphere ~$ ls
apcb      Documents Music      Public      todo_list.txt
Desktop  Downloads Pictures  Templates  Videos
```

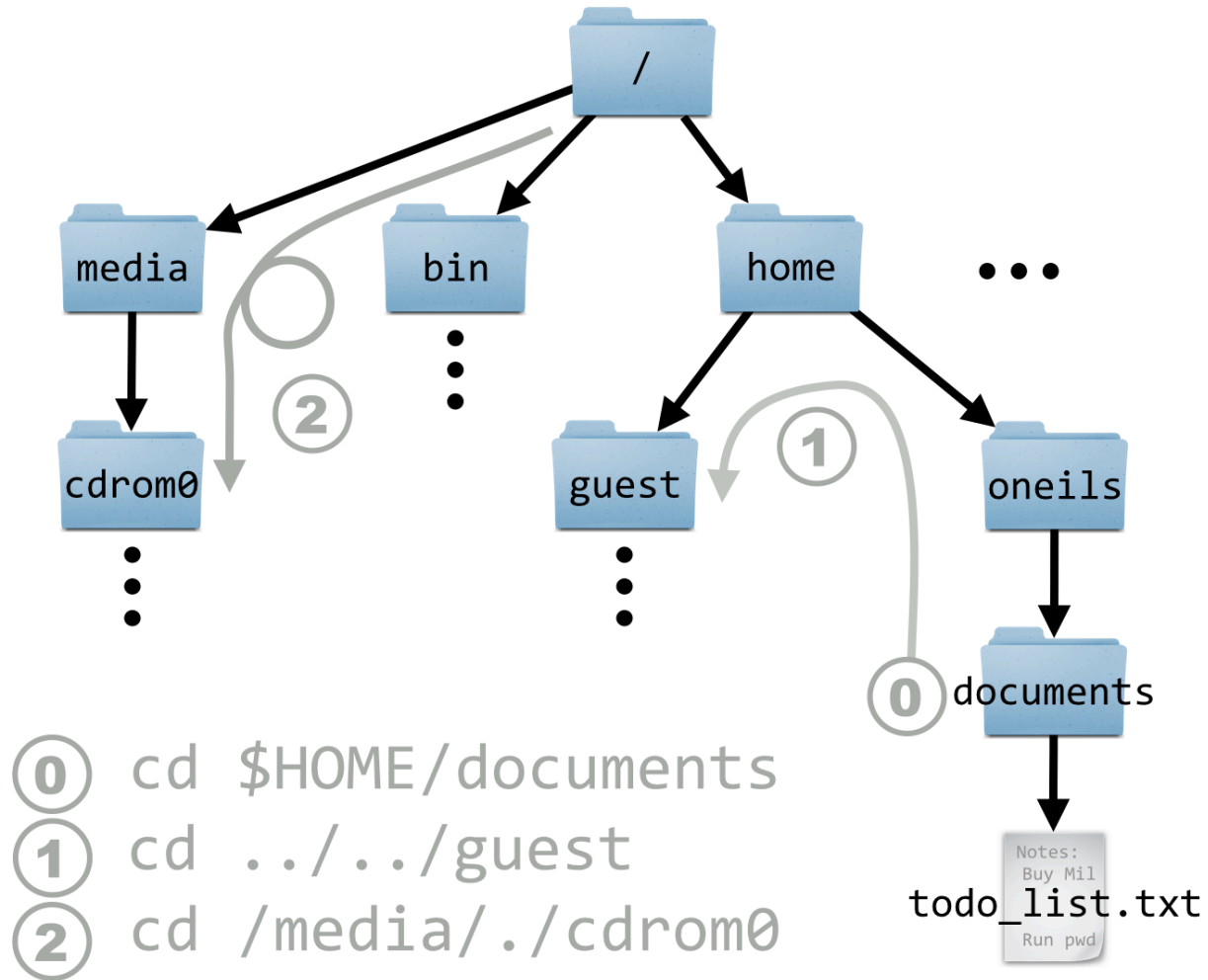
- On peut **changer de répertoire** (directory) par la commande **cd** (change directory)

```
oneils@atmosphere ~$ cd /home
oneils@atmosphere /home$ echo $PWD
/home
oneils@atmosphere /home$ ls
lost+found oneils
```

- On peut **revenir** à son « home directory » par :

- cd \$HOME
- cd
- cd ~

Se déplacer dans la hiérarchie des répertoires



...

Chemins **absolus** et chemins **relatifs**

Chemin absolu : `/home/oneils/Pictures/profile.jpg`

Chemin absolu : `/home/oneils/todo_list.txt`

- Supposons être **dans l'home directory**
 - Chemin relatif : `oneils/Pictures/profile.jpg`
 - Chemin relatif : `oneils/todo_list.txt`
- Supposons être **dans home/oneils**
 - Chemin relatif : `Pictures/profile.jpg`
 - Chemin relatif : `todo_list.txt`

Remarque :

- Les chemins **absolus** commencent par /
- Pas les chemins **relatifs**

Remarques

- Dans les systèmes Unix, les commandes sont **sensibles à la casse**
- Les espaces comptent
 - On donnera donc des **noms** de fichiers et de répertoires **sans espaces**
 - Eg. `todo_list.txt`
 - Sinon il faut utiliser `'todo list.txt'` ou `todo\list.txt` dans les commandes (à **éviter !!!**)

Les fichiers cachés

- Par défaut les fichiers commençant par `.` sont « cachés »
- Il faut utiliser l'option `-a` pour les voir

```
oneils@atmosphere ~/Pictures$ cd $HOME
oneils@atmosphere ~$ ls -a
.                .config          .gvfs             .profile          .vim
..               .dbus            .gvfs             Public            .viminfo
apcb             Desktop          .ICEauthority     .pulse            .vimrc
.bash_history    Documents        .local            .pulse-cookie    .vnc
.bash_login      Downloads        Music              .ssh              .Xauthority
.bash_logout     .gconf           .netrc            Templates         .Xdefaults
.bashrc          .gnome2          Pictures           todo_list.txt     .xscreensaver
.cache           .gnupg           .pip              Videos           .xsession-errors
```

- Ce sont en général des **fichiers de configuration** utilisés par des programmes variés

Commande **ls** et ses **options**

- `ls` affiche la liste des fichiers et sous-répertoires du **répertoire courant**
- `ls rep1/toto` affiche la liste des fichiers et sous-répertoires du **répertoire rep1/toto**
- `ls -l` affiche une **liste détaillée** (droit, propriétaire, taille, etc.)
- `ls -a` affiche aussi les **fichiers cachés**
- `ls -t` affiche **par ordre de date** de dernière modification

...

- Option **-h** pour « human readable »
 - Donne les tailles mémoire en Ko ou Mo

```
oneils@atmosphere ~$ ls -lah
total 168K
drwxr-xr-x 25 oneils iplant-everyone 4.0K Sep 23 22:40 .
drwxr-xr-x  4 root    root          4.0K Sep 15 09:48 ..
drwxr-xr-x  4 oneils iplant-everyone 4.0K Sep 15 11:19 apcb
-rw-----  1 root    root          2.2K Sep 15 10:49 .bash_history
-rw-r--r--  1 oneils iplant-everyone   61 Sep 16 19:46 .bash_login
-rw-r--r--  1 oneils iplant-everyone  220 Apr  3 2012 .bash_logout
-rw-r--r--  1 oneils iplant-everyone  3.6K Sep 15 09:48 .bashrc
drwx-----  7 oneils iplant-everyone 4.0K Sep 15 09:52 .cache
...
```


- réfère au répertoire courant
- réfère au répertoire home
 - E.g. **cd ..** remonte au répertoire home

```
oneils@atmosphere ~$ echo $PWD
/home/oneils
oneils@atmosphere ~$ cd .
oneils@atmosphere ~$ echo $PWD
/home/oneils
oneils@atmosphere ~$ cd ..
oneils@atmosphere /home$ echo $PWD
/home
```

Créer de nouveaux répertoires


mkdir (make directory)

```
oneils@atmosphere ~$ ls
apcb      Documents  Music      Pictures    Templates  Videos
Desktop  Downloads  p450s.fasta Public      todo_list.txt
oneils@atmosphere ~$ mkdir projects
oneils@atmosphere ~$ ls
apcb      Documents  Music      Pictures    Public      todo_list.txt
Desktop  Downloads  p450s.fasta projects    Templates  Videos
```



Copier ou renommer un fichier ou un répertoire

mv (move)



```
oneils@atmosphere ~$ mv p450s.fasta p450s.fa
oneils@atmosphere ~$ mv p450s.fa projects
oneils@atmosphere ~$ mv projects projects_dir
oneils@atmosphere ~$ ls
apcb      Documents  Music      projects_dir  Templates    Videos
Desktop  Downloads  Pictures    Public        todo_list.txt
oneils@atmosphere ~$
```

- Si la **destination** n'existe pas, elle est **créée**
- Si la **destination** existe
 - Si c'est un répertoire, la **source est déplacée** dans celui-ci
 - Si c'est un fichier, celui-ci est **écrasé (!!!)** et le contenu remplacé par celui de la source

Copier un fichier ou un répertoire

cp (copy)

- Avec l'option `-r` (récursif) si l'on veut copier tout le contenu d'un répertoire avec ses sous-répertoires et ses fichiers

```
oneils@atmosphere ~$ cp todo_list.txt todo_copy.txt  
oneils@atmosphere ~$ cp -r projects projects_dir_copy
```

Retirer ou effacer un fichier ou un répertoire

rm (remove)

```
oneils@atmosphere ~/projects$ mkdir tempdir
oneils@atmosphere ~/projects$ ls
p450s.fasta  tempdir  todo_list.txt
oneils@atmosphere ~/projects$ rm todo_list.txt
oneils@atmosphere ~/projects$ rm -rf tempdir/
oneils@atmosphere ~/projects$ ls
p450s.fasta
```

- **ATTENTION !!!**
 - Les fichiers ou répertoires effacés **ne peuvent pas être récupérés (!!!)**
 - Pas de Ctrl z
 - Pas de « corbeille »

Caractères **spéciaux** utiles

- **?** : remplace **un caractère** quelconque

- `mv ../data/out0?.dat ~/poub/`

- déplace tous les fichiers `.dat` de nom commençant par `out0` avec un caractère de plus du répertoire `../data` dans le répertoire `poub` du répertoire personnel (indiqué par le caractère spécial `~`) (e.g. `home/dupont`)

- ***** : remplace une **chaîne de caractères** quelconque

- `rm rep1/*.dat` : détruit tous les fichiers du répertoire `rep1` dont le nom fini par `.dat`

Miscellanées

- Tab completion
- Obtenir de l'aide sur une commande
 - E.g. `man ls`
- Voir quels sont les programmes qui tournent : `top`
 - Fournit le pourcentage de CPU consommé, combien de mémoire, les utilisateurs, ...
 - On quitte par `q`

```
top - 02:23:20 up 15 days, 16:34, 2 users, load average: 0.03, 0.02, 0.05
Tasks: 127 total, 1 running, 126 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.0%us, 0.0%sy, 0.0%ni, 99.7%id, 0.3%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4050124k total, 1801608k used, 2248516k free, 142652k buffers
Swap: 0k total, 0k used, 0k free, 1256080k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	90540	4264	2696	S	0.0	0.1	0:07.85	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	0:43.50	ksoftirqd/0
5	root	20	0	0	0	0	S	0.0	0.0	0:00.35	kworker/u:0
6	root	RT	0	0	0	0	S	0.0	0.0	0:00.00	migration/0

Obtenir de l'information sur les utilisateurs

finger

```
[(base) MacBook-Pro-de-ANTOINE-5:Man_Unix antoincornuejols$ finger antoine
Login: antoincornuejols                Name: ANTOINE CORNUEJOLS
Directory: /Users/antoincornuejols     Shell: /bin/zsh
On since Mer 23 août 17:41 (CEST) on console, idle 5 days 1:44 (messages off)
On since Mer 23 août 18:19 (CEST) on ttys000, idle 5 days 0:39
On since Mer 23 août 18:47 (CEST) on ttys001, idle 4 days 1:35
On since Jeu 24 août 17:51 (CEST) on ttys002
On since Mer 23 août 18:02 (CEST) on ttys003 (messages off)
No Mail.
No Plan.
(base) MacBook-Pro-de-ANTOINE-5:Man_Unix antoincornuejols$
```

Les permissions

- Tous les fichiers et répertoires sont associés à un **utilisateur** (le propriétaire) et un **groupe**, plus « **les autres** ».
- Les **permissions** déterminent ce que peuvent faire **l'utilisateur**, le **groupe** et les **autres**.
- Elles sont décrites par une **combinaison de permissions** de :
 - Lecture (r : read)
 - Écriture (w : write)
 - Exécution (x : execute)

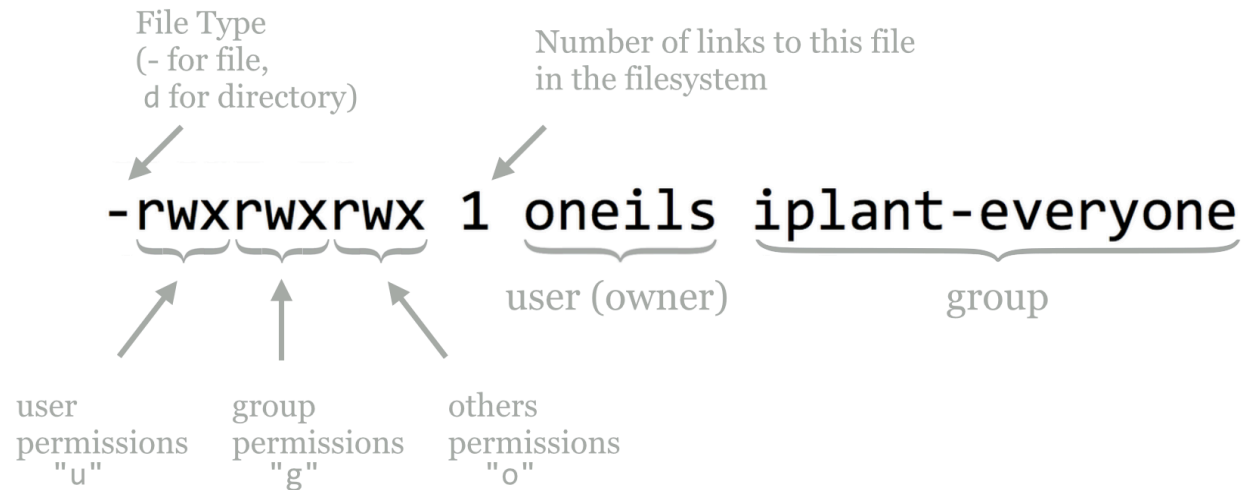
Les permissions

- Exemple :

```
oneils@atmosphere ~/apcb/intro$ ls -l
total 20
-rwxrwxrwx 1 oneils iplant-everyone 15891 Oct 20 17:42 p450s.fasta
drwxr-xr-x 2 oneils iplant-everyone  4096 Oct 20 17:40 temp
```

- Pour le 1er fichier : la combinaison `rwxrwxrwx` permet à tout le monde de tout faire
- Pour le répertoire :
 - Est accessible par **tout le monde** : X
 - Peut être lu par les membres du **groupe** : rX
 - Et modifié seulement par le **propriétaire** : rWX

Les permissions

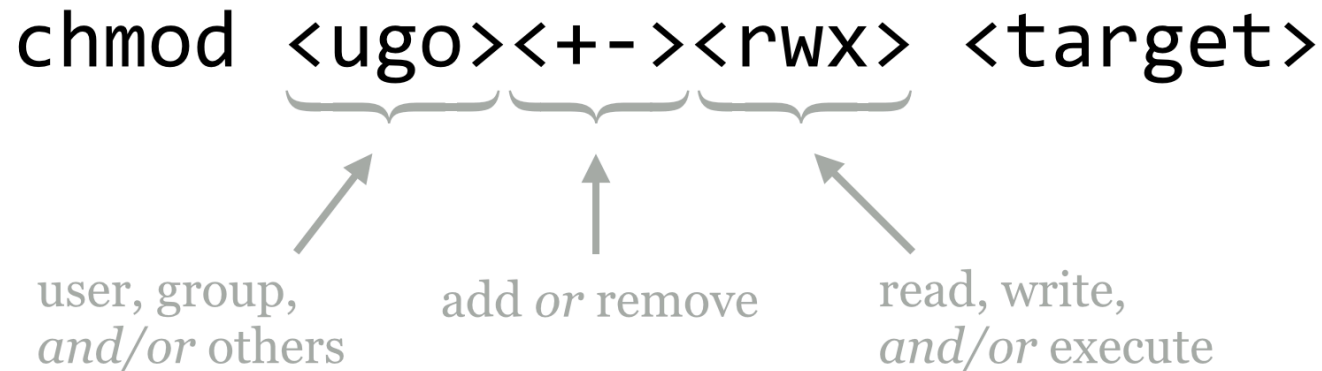


Code	Meaning for Files
r	Can read file contents
w	Can write to (edit) the file
x	Can (potentially) "execute" the file

Code	Meaning for Directories
r	Can see contents of the directory (e.g., run <code>ls</code>)
w	Can modify contents of the directory (create or remove files/directories)
x	Can <code>cd</code> to the directory, and potentially access subdirectories

...

Les modifications de permissions



Exemples :

Command	Effect
<code>chmod go-w p450s.fasta</code>	Remove write for group and others
<code>chmod ugo+r p450s.fasta</code>	Add read for user, group, and others
<code>chmod go-rwx p450s.fasta</code>	Remove read, write, and execute for group and others
<code>chmod ugo+x p450s.fasta</code>	Add execute for user, group, and others
<code>chmod +x p450s.fasta</code>	Same as <code>chmod ugo+x p450s.fasta</code>

Exercices

1. Dans votre home, tapez `pwd`. Qu'est-ce qui est retourné ?
2. Créez un répertoire `Man_Shell` dans `home` et déplacez vous dedans
3. Tapez `pwd`. Qu'est-ce qui est retourné ?
4. Créez un fichier `test.txt` dans `home/Man_Shell`
5. Grâce à un éditeur de texte, tapez quelques lignes dans ce fichier
6. Quelles sont les permissions par défaut données au fichier ?
7. Modifiez ces permissions pour le rendre lisible par le groupe
8. Copiez ce fichier dans un fichier `test2.txt` dans `home/Man_Shell`

Commandes Bash

- `pwd` **print working directory**
 - Renvoie le répertoire du chemin courant
- `mkdir` **make dir**
 - Permet de créer de nouveaux répertoires
- `chmod`
 - Change les permissions d'accès aux fichiers
- `ls` **list files and directories**
- `cd` **change directory**
- `file` retourne le type de fichier
- `compgen` (avec un pipe serait bien)
 - Liste des différentes commandes disponibles
- `man`
 - Fournit une description de la commande fournie en argument

Commandes Bash de manipulation de fichiers

- cp **copy file**
- mv **move**
 - déplace ou renomme des fichiers
- rm **remove**
 - Supprime ou renomme des fichiers ou des répertoires

Les commandes Bash

et le contenu des fichiers

Comparer des fichiers grâce à **diff**

- Créez fichier_1.txt avec dedans le texte « Ceci est une chaine de caractères »
- Copiez fichier_1.txt dans fichier_2.txt
 - Que donne `$ diff fichier_1.txt fichier_2.txt ?`
 - Et `$ diff -s fichier_1.txt fichier_2.txt ?`
- Modifiez le contenu de fichier_2.txt en remplaçant des minuscules par des majuscules
 - Que donne `$ diff fichier_1.txt fichier_2.txt ?`
 - Que donne `$ diff fichier_1.txt fichier_2.txt -i ?`
 - Pourquoi ?

Comparer des fichiers grâce à **diff**

- Modifiez le contenu de fichier_1.txt et de fichier_2.txt en ajoutant respectivement « ligne 1 » et « ligne 2 »
- Que donne maintenant `$ diff fichier_1.txt fichier_2.txt` ?

Pour en **savoir davantage** utilisez `diff --help`

Commandes Bash

wc (word count) pour compter

- `wc -c` : compte le nombre de **caractères** du fichier
- `wc -w` : compte les **mots** du fichier
- `wc -l` : compte le nombre de **lignes** (en fait les newlines) du fichier

```
$ wc -l programming.txt
10 programming.txt
```

```
$ cat programming.txt | wc -l
10
```

pipe



```
$ ls -l | wc -l
1184
```

Nombre de lignes (donc de fichiers)
dans le répertoire courant

Commandes Bash

wc (word count) pour compter

- Compter le **nombre d'apparitions d'un mot** dans un fichier est **complexe**.
- Compter le **nombre de lignes où il apparaît** est plus **facile**

```
$ cat notes | grep the | wc -l
32
$ cat notes | grep [Tt]he | wc -l
40
```

Nombre de lignes avec le mot « the »

Nombre de lignes avec le mot
« the » ou « The »

Exercices

9. Quelle commande devez-vous utiliser pour avoir la liste des fichiers de `home/Man_Shell` ?
10. Copiez maintenant le fichier `microbiome.txt` disponible sur e-campus
11. Affichez les 5 premières lignes de `microbiome.txt`.
12. Affichez les 4 dernières lignes de `microbiome.txt`.
13. Comptez le nombre de lignes de `microbiome.txt`.

Exercices

14. Recherchez tous les chiffres dans `microbiome.txt` et les afficher
15. Combien y en a-t-il ?
16. Comptez le nombre de 'e' dans le texte

Commande cat

- de l'anglais *catenate*, synonyme de **concatenate** (concaténer), est une commande Unix standard permettant de
 - **concaténer** des fichiers
 - ainsi que **d'afficher leur contenu** sur la sortie standard
- `cat fichier.txt`
 - Affiche le contenu de `fichier.txt`
- `cat fichier.txt | more`
 - Affiche la 1^{ère} page de `fichier.txt` et plus de lignes si on fait `return`
- `cat -n fichier.txt`
 - Affiche le contenu de `fichier.txt` en numérotant les lignes (voir `cat -b` qui ne numérote que les lignes non vides)
- `cat fichier1.txt fichier2.txt`
 - Affiche le contenu des deux fichiers
- `cat source1.txt source2.txt > destination.txt`
 - Met le contenu des fichiers `source1.txt` et `source2.txt` dans le fichier `destination.txt`
- `cat -v nomdufichier.txt`
 - Affiche les caractères non imprimables du fichier

Commande grep

- `grep [options] <expression> <fichiers/répertoires>`
 - **grep** cherche la chaîne de caractères <expression> à l'intérieur des fichiers ou des répertoires spécifiés et affiche les lignes correspondantes.
 - Cette commande permet l'utilisation d'expressions régulières (voir plus loin)

Commande grep

- La commande **grep** permet de capturer un motif dans un texte (Global Regular Expression Parser)

- `grep "Paris" fichier`

- Affiche toutes les lignes de fichier qui contiennent « Paris »

- `grep -c "Paris" fichier`

- Affiche le nombre de lignes de fichier qui contiennent « Paris »

- `grep "^Paris" fichier`

- Affiche les lignes de fichier qui commencent par « Paris »
(le méta caractère ^ signifie « qui commence par »)

- `grep "Paris" repertoire1/*`

- Affiche la liste des fichiers de repertoire1 qui contiennent « Paris »

- `grep -v "Paris" fichier`

- Affiche toutes les lignes de fichier qui **ne** contiennent **pas** « Paris »

- `grep -i -v "Paris" fichier`

- Affiche toutes les lignes de fichier qui **ne** contiennent **pas** « Paris » (-i signifie ne pas faire attention à la casse)

Commande grep

- `grep -A3 "Paris" fichier`
 - Affiche 3 lignes de fichier **après** (After) le mot « Paris »
- `grep -B2 "Paris" fichier`
 - Affiche 2 lignes de fichier **avant** (Before) le mot « Paris »
- `grep -C5 "Paris" fichier`
 - Affiche 5 lignes de fichier **avant et après** le mot « Paris »
- `grep -e "Paris" -e "Londres" fichier`
 - Affiche toutes les lignes de fichier qui contiennent « Paris » ou « Londres »
- `grep -E "Paris|Londres" fichier`
 - Idem en utilisant une expression régulière (voir plus loin)
- `Grep "^#" fichier`
 - Pour afficher les lignes commençant par # (donc les lignes de commentaires)
- `Grep "\.$" fichier`
 - Pour afficher les lignes se terminant par un point (\$ signifie se terminant)