

# | Linux - Terminal

Guillaume Rodrigues  
17/06/2024

## Table des matières

Introduction au Shell .....	3
Shell ça veut bien dire coquillage ? .....	3
Les commandes .....	4
Les Scripts .....	5
Challenge.....	6
Ressources .....	6
Terminal : Arborescence .....	7
Introduction .....	7
Utilisateurs Mac et Windows.....	7
MacOS .....	7
Windows.....	7
Le terminal .....	8
A quoi ça sert ?.....	8
Un peu d'histoire .....	8
L'arborescence des répertoires sous Linux .....	10
Explorer l'arborescence avec le terminal .....	11
Lancer le terminal .....	11
L'invite de commande ou "prompt" .....	12
Lister le contenu d'un répertoire avec ls .....	13
Afficher le répertoire courant avec 'pwd' .....	14
Changer de répertoire avec cd .....	14
Signification de ~ .....	14
Pour en savoir plus sur les commandes de base .....	14
Terminal : Fichiers et dossiers .....	15
Introduction .....	15
Créer des répertoires .....	15
Supprimer des répertoires .....	15
Créer des fichiers.....	15
Copier des fichiers .....	16
Exemple 1 : Copier un fichier .....	16

Exemple 2 : Copier plusieurs fichiers dans un répertoire .....	17
Exemple 3 : Faire une copie récursive.....	17
Exemple 1 : Supprimer les doublons .....	19
Exemple 2 : Supprimer récursivement un répertoire .....	19
Déplacer et/ou renommer .....	19
Pour aller plus loin .....	19
Challenge.....	20
Télécharger le fichier.....	20
Exercice.....	21

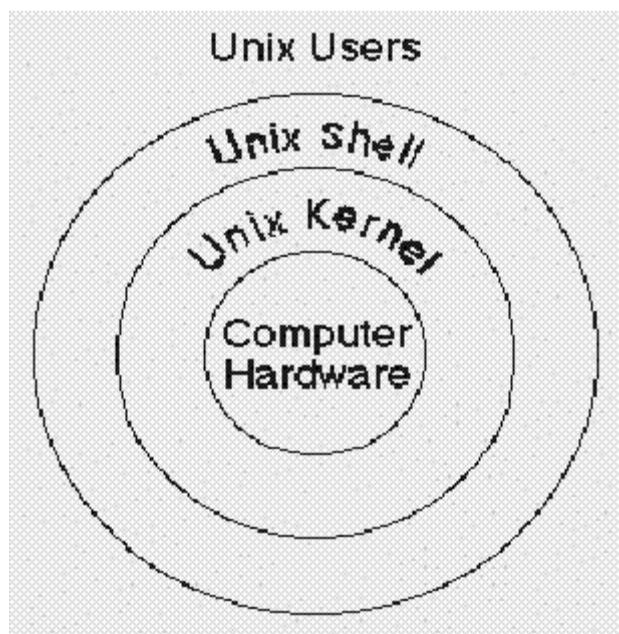
# Introduction au Shell

Le shell, c'est ce fameux langage qui permet d'interagir avec une machine sans interface graphique. Un peu repoussant au premier abord, mais tellement pratique et efficace une fois que l'on sait s'en servir.

Les commandes du shell se saisissent dans un terminal qu'on appelle aussi console, ou invite de commande.

## Shell ça veut bien dire coquillage ?

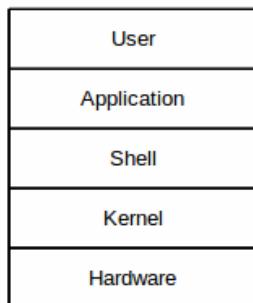
En effet, shell est le mot anglais pour coquillage ou coquille. Et c'est le rôle qu'il joue dans l'architecture UNIX : Il protège et sublime le système.



Dans ce schéma, tu peux voir que le shell se situe entre le kernel (le noyau) de l'ordinateur et l'utilisateur. Ce type d'architecture en couches est très présent dans les systèmes informatiques, car il permet de les décomplexifier et laisse les experts se spécialiser sur une couche.

Introduction aux scripts Shell : [https://doc.ubuntu-fr.org/tutoriel/script\\_shell#:~:text=Un%20script%20shell%20permet%20d,seront%20ex%C3%A9cut%C3%A9s%20de%20mani%C3%A8re%20quentielle.](https://doc.ubuntu-fr.org/tutoriel/script_shell#:~:text=Un%20script%20shell%20permet%20d,seront%20ex%C3%A9cut%C3%A9s%20de%20mani%C3%A8re%20quentielle.)

On représente souvent les architectures par couches de cette façon :



Ainsi on appelle les couches d'en bas les couches de bas niveau, et les couches du haut les couches de haut niveau. C'est fou, non ? Ainsi, quand tu entends parler d'un langage de bas niveau, ça ne veut pas dire qu'il est pour les nuls mais qu'il se rapproche du langage machine.

Tu remarqueras que, dans le second schéma, j'ai rajouté une couche pour les applications. En général les utilisateurs n'utilisent pas directement le shell, mais des applications que les développeurs ont créées pour eux et qui elles-mêmes utilisent le shell. Les applications avec interface graphique sont plus intuitives, conviviales et rassurantes pour l'utilisateur, mais le brident dans son utilisation de la machine.

Tu l'auras compris, apprendre à utiliser le shell, c'est prendre le contrôle de l'ordinateur.

## Les commandes

Un langage met à disposition de celui qui l'utilise un ensemble de commandes. Le shell n'échappe pas à la règle. Nous allons découvrir ensemble comment s'en servir.

La première chose que tu dois trouver, c'est un interpréteur qui comprend le shell. Sous Linux, tu peux ouvrir un terminal en pressant Ctrl+Alt+t, sous Mac tu trouveras un terminal dans les applications (Mac est basé sur un noyau UNIX).

Quelle est la différence entre Unix, Linux et Ubuntu ?

Ubuntu est une distribution du système d'exploitation Linux. Linux est construit sur un noyau Unix. Tu te souviens du schéma de l'étape précédente ? Eh bien les systèmes d'exploitation sont souvent livrés avec un ensemble d'applications qui caractérisent une distribution.

Une fois que tu es dans ton terminal, essaie les commandes suivantes une par une.

```
pwd  
ls  
cd ~  
mkdir files  
cd files  
cat > hello.txt  
Bonjour Madame  
Ctrl+c  
ls  
cat hello.txt
```

Maintenant commentons. La commande pwd affiche le dossier courant, c'est le dossier dans lequel on se trouve.

ls permet d'afficher les dossiers et fichiers du dossier courant

cd veut dire change directory. Cette commande permet de changer le dossier courant

mkdir veut dire make directory. Elle parle d'elle-même, elle permet de créer un dossier dans le dossier courant.

La commande cat > hello.txt permet d'écrire dans le fichier hello.txt jusqu'à ce que tu pousses Ctrl-c

Si tu tapes cat hello.txt, ça affiche le contenu du fichier hello.txt

Pour finir le tour des commandes incontournables, je t'invite à jeter un œil ici :

<http://mally.stanford.edu/~sr/computing/basic-unix.html>

## Les Scripts

Un script est un ensemble de commandes regroupées dans un fichier. En fait, je suis en train de te dire que tu peux créer tes propres commandes à partir des commandes fournies par le shell.

Par exemple, dans ton terminal, tape les lignes suivantes :

```
cat > wcs.sh  
#!/bin/bash  
echo "-----|-----|-----|-----|"  
echo "| | ____| | | | | | | | | | | | | | | |"  
echo "| | | | | | | | | | | | | | | | | | | | | |"  
echo "| | | | | | | | | | | | | | | | | | | | | |"  
echo "| | | | | | | | | | | | | | | | | | | | | | |"  
echo "| | | | | | | | | | | | | | | | | | | | | | | |"  
echo "| | | | | | | | | | | | | | | | | | | | | | | | |"  
echo "| | | | | | | | | | | | | | | | | | | | | | | | | |"  
echo "-----|-----|-----|-----|"  
Ctrl+c
```

Tu viens de créer ton premier script qui affiche WCS en ASCII art. Pour le tester, lance la commande :

```
./wcs.sh
```

Si tu as un problème de permission, donne les droits en exécution au fichier avec la commande suivante :

```
chmod +x wcs.sh
```

## Challenge

Je te propose pour ce challenge de créer un script qui salue l'utilisateur. Le script bonjour.sh prend en paramètre le nom de la personne à saluer. On devra avoir le retour suivant :

```
$ ./bonjour.sh Romain  
bonjour Romain
```

Crée un gist avec le fichier bonjour.sh et poste en solution un lien vers ce gist.

Bonus : Si aucun paramètre n'est saisi, le script saluera l'utilisateur connecté.

## Ressources

<https://mally.stanford.edu/~sr/computing/basic-unix.html>

[https://doc.ubuntu-fr.org/tutoriel/script\\_shell#:~:text=Un%20script%20shell%20permet%20d,seront%20ex%C3%A9cut%C3%A9es%20de%20mani%C3%A8re%C3%A8re%20s%C3%A9quentielle.](https://doc.ubuntu-fr.org/tutoriel/script_shell#:~:text=Un%20script%20shell%20permet%20d,seront%20ex%C3%A9cut%C3%A9es%20de%20mani%C3%A8re%C3%A8re%20s%C3%A9quentielle.)

# Terminal : Arborescence

## Introduction

Le développeur doit souvent utiliser le terminal pour travailler sur son ordinateur afin d'effectuer diverses opérations.

Le terminal peut sembler effrayant à première vue.

Mais tu verras, c'est un outil vraiment facile à utiliser et il peut t'aider à effectuer certaines opérations de base plus rapidement.

Dans cette quête, tu vas découvrir le terminal, ainsi que quelques commandes de base pour explorer l'arborescence des répertoires.

## Utilisateurs Mac et Windows

Le terminal que tu vas découvrir, dans cette quête et les suivantes, est un héritage du système d'exploitation Unix... Dont Linux est un descendant.

Et si tu n'utilises pas Linux, mais MacOS ou Windows ?

### MacOS

Bonne nouvelle : MacOS descend également d'Unix, et il est livré avec un terminal (recherche Spotlight via cmd+space, puis "terminal").

La principale différence avec Linux est que ton répertoire personnel sera quelque chose comme /Users/myname au lieu de /home/myname.

D'autres différences plus subtiles concernent les options acceptées par certaines commandes, mais cela ne devrait pas t'affecter lors de ces quêtes "terminal".

### Windows

Windows est un descendant de MS-DOS, qui s'est lui-même inspiré d'Unix.

Il fournit un terminal, offrant des équivalents aux commandes Unix, mais avec des noms différents !

Il est donc nécessaire d'installer un autre terminal, afin de bénéficier d'un environnement proche des systèmes Unix.

Tu devras installer Git Bash.

Veinard, en faisant cela, tu auras déjà Git installé !

Une fois le processus d'installation terminé, Tu pourras suivre ces quêtes sans problème, la principale différence étant que ton répertoire d'origine sera quelque chose comme /c/Users/MyName au lieu de /home/myname.

## Le terminal

### A quoi ça sert ?

Le terminal te permet d'exécuter des commandes en les tapant sur le clavier.

Quel intérêt à une époque où nos ordinateurs ont des interfaces graphiques conviviales et nos smartphones des écrans tactiles ?

Eh bien, même si ce n'est pas immédiatement évident pour toi, le terminal offre de nombreuses possibilités. Par exemple, il y a des commandes pour :

- Manipuler (copier, déplacer, renommer, supprimer) des fichiers et des répertoires
- Extraire des informations spécifiques à partir de fichiers texte volumineux plus rapidement qu'avec un éditeur de texte.
- Rechercher efficacement des fichiers dans une arborescence de fichiers, selon différents critères (nom, date de modification, etc.)
- Installer un logiciel
- et d'autres choses (la liste complète serait longue !)

Avec un peu de pratique, certaines opérations se font beaucoup plus rapidement dans le terminal que via l'interface utilisateur graphique (GUI) !

### Un peu d'histoire

Le mot terminal désignait à l'origine un "ordinateur" rudimentaire, équipé d'un simple clavier, et relié par un réseau à un serveur central.

Ci-dessous, un des précurseurs, le Dec VT100.



L'utilisateur se connecte au serveur et peut alors entrer des commandes à l'aide du clavier.

Bien que la souris ait été inventée en même temps que les terminaux, les ordinateurs n'en étaient pas équipés et, de plus, les écrans étaient très limités et ne pouvaient afficher que du texte, en vert sur un fond noir (penses-y, la prochaine fois que tu regarderas The Matrix !)



## L'arborescence des répertoires sous Linux

Sur tous les systèmes d'exploitation, les fichiers et les répertoires sont organisés en "arborescence", c'est-à-dire une structure hiérarchique.

Le terme "arborescence" n'est pas une coïncidence, car une arborescence de fichiers ressemble à un bon vieil arbre : la "racine" de l'arborescence serait le tronc, les répertoires, ses branches, et les fichiers, ses feuilles.

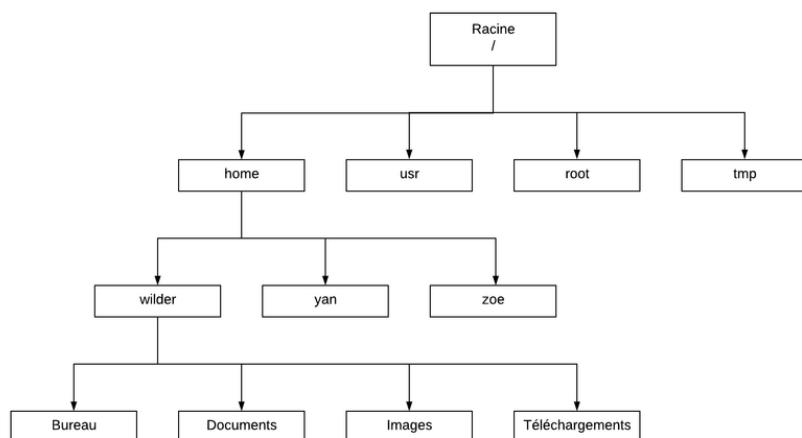
Il existe certaines disparités entre les systèmes : par exemple, sous Windows, chaque lecteur physique (partition de disque dur, lecteur optique de DVD ou Blu Ray, clé USB) se voit attribuer une lettre (C : pour le disque dur de démarrage, D : pour le lecteur optique, par exemple).

Ce n'est pas le cas sous les systèmes Unix, dont Linux et OS X font partie. Sous ces systèmes, tous les disques apparaissent sous la même arborescence.

Utilisateurs de Windows : Git Bash émule en fait cette arborescence unifiée. Votre disque C: apparaîtra sous la forme /c sous Git Bash.

Voyons maintenant à quoi ressemble cette arborescence sous Linux.

L'image ci-dessous n'en montre qu'une très petite partie car pour la montrer en entier, tu auras probablement besoin d'un écran de 450 pouces !



## Explorer l'arborescence avec le terminal

Nous entrons dans le vif du sujet !

Commence par regarder cette vidéo

(<https://www.youtube.com/watch?v=AO0jzD1hpXc>), qui montre comment utiliser le terminal, et quelques commandes de base.

Conseil général, ne suis pas passivement la vidéo : tape toi-même les commandes dans le terminal !

Ce qui suit est un complément, largement redondant avec la vidéo, mais avec des exemples supplémentaires.

Suis la vidéo, garde le terminal ouvert et n'efface rien (pas de "clear") après avoir essayé les différentes commandes.

Cela te servira pour le challenge.

## Lancer le terminal

Le terminal est l'une des nombreuses applications fournies par défaut avec Linux.

Pour clarifier : parler du terminal n'est pas correct, il serait plus exact de parler de l'émulateur de terminal.

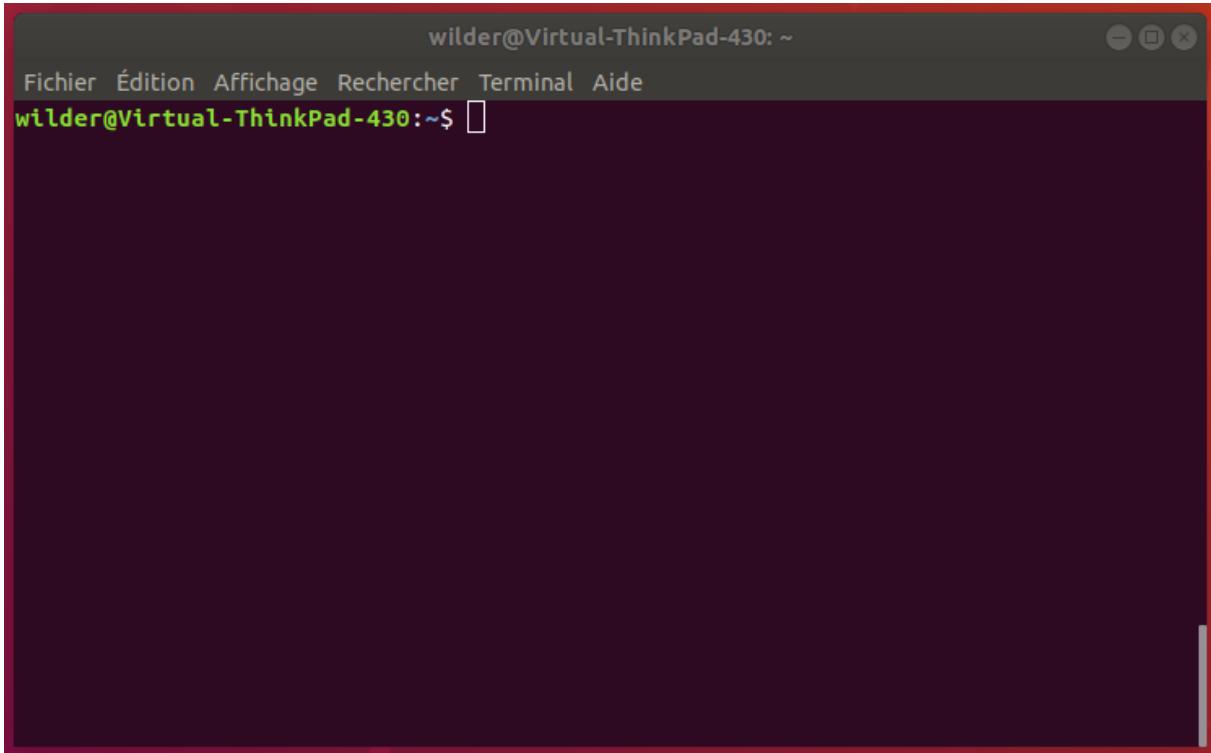
Pour lancer le terminal, tu devras d'abord le trouver !

Tu pourras le trouver en haut à gauche de l'écran sous Ubuntu.

- Clique sur "Activités",
- Tape "ter", pour trouver l'application du terminal.
- Une fois le terminal lancé, si tu souhaites ajouter une nouvelle icône, fais un clic droit sur son icône dans la barre latérale gauche, puis clique sur "Ajouter aux favoris", afin qu'elle apparaisse toujours dans la barre des favoris !

## L'invite de commande ou "prompt"

L'émulateur de terminal affiche une fenêtre avec un fond sombre.



Dans cette fenêtre, un curseur clignotant est affiché, précédé de l'invite de commande ou prompt en anglais. L'invite de commande peut être décomposée comme suit :

- La première partie, en vert, username@Computer-Name indique qui nous sommes et sur quelle machine nous travaillons. Ici, nous travaillons localement, mais nous pourrions très bien nous connecter à distance à une autre machine Linux, auquel cas cette partie changerait.
- La deuxième partie en bleu, séparée de la première par une flèche, indique où tu te trouves dans l'arborescence... Nous reviendrons sur la signification de ~ plus tard.
- Le dernier caractère, \$... finissant l'invite.

Après l'invite se trouve le curseur, et maintenant c'est à toi de jouer !

Tu vas te faire la main sur quelques commandes. Un conseil : **essaye-les au fur et à mesure !**

## Lister le contenu d'un répertoire avec ls

La commande 'ls' te permet d'afficher le contenu d'un répertoire.

Quelques exemples, en supposant que tu viens de lancer le terminal :

- 'ls' affiche les fichiers et répertoires contenus dans le répertoire courant, celui dans lequel tu es actuellement.
- 'ls .' fait la même chose ! Le symbole . signifie "le répertoire actuel".
- 'ls -a' affiche le contenu du répertoire, mais en incluant les fichiers cachés. Ceux-ci sont précédés du caractère . pour les différencier des fichiers "normaux". Attention, ici, le . est annexé au nom du fichier, il n'a pas la même signification que le symbole . pris isolément.
- 'ls Images' et 'ls ./Images' affiche le contenu du dossier Images.
- 'ls /home/username' est une autre façon d'afficher le contenu de ton répertoire personnel, qui se trouve être le répertoire par défaut au lancement du terminal.
- 'ls /home' te permet d'afficher la liste des répertoires associés à chaque utilisateur du système.
- 'ls ..' te donnera le même résultat ! Le .. signifie le répertoire parent. Celui qui se trouve juste au-dessus dans l'arborescence. /home est au-dessus de /home/username.
- 'ls /bin' affiche le contenu d'un répertoire contenant les commandes de base du système (par exemple, la commande ls elle-même !).
- 'ls -l /bin' fait la même chose, mais donne plus de détails, comme par exemple qui possède les fichiers, quand ils ont été modifiés pour la dernière fois, et ainsi de suite.

Je voudrais profiter de cette occasion pour introduire ici la notion de paramètres ou d'arguments d'une commande : c'est simplement tout ce qui vient après la commande elle-même.

- Pour prendre le dernier exemple, -l et /bin sont les arguments donnés à ls.
- Les arguments commençant par - ou -- comme -a, -l ou --help sont appelés flags ou options ; ils modifient le comportement de la commande.

Autre chose, qui peut sembler "évidente" : le caractère slash (/) permet d'enchaîner les noms des répertoires et de leurs sous-répertoires.

C'est le séparateur de répertoires.

## Afficher le répertoire courant avec 'pwd'

La commande 'pwd' te permet d'afficher le répertoire courant, également appelé "répertoire de travail". D'où le nom de la commande, abréviation de print working directory.

Lorsque tu viens de lancer le terminal, pwd devrait t'afficher /home/username.

## Changer de répertoire avec cd

L'ordre cd signifie changer de répertoire et te permet donc de changer ton répertoire de travail.

Quelques exemples, à essayer dans l'ordre :

- cd Images : se déplacer sous le dossier Images (vérifier avec pwd).
- cd .. : remonter dans le répertoire parent/home/username.
- cd /opt : aller sous un répertoire en précisant son chemin absolu. L'emplacement de l'arborescence, c'est-à-dire sa position par rapport à la racine de l'arborescence.
- cd .. /home/username: d'abord, retourne à la racine (via ..) et de là, retourne au fichier /home/username.
- cd /usr/bin : aller dans un autre répertoire, celui-ci étant un autre répertoire contenant des commandes (vérifier avec ls).
- cd sans paramètres : retour au répertoire personnel de l'utilisateur courant.
- cd ~ Même chose.
- cd ~/Music : va sous ton dossier Music (quel que soit le dossier dans lequel tu étais avant).

## Signification de ~

Le caractère ~ (appelé tilde) représente le répertoire personnel de l'utilisateur actuel.

C'est un moyen rapide d'accéder à l'adresse /home/username (remplacer le username par celui de l'utilisateur actuel).

C'est très pratique, surtout si tu es dans un répertoire complètement différent et que tu veux consulter un élément de ton dossier personnel.

Par exemple, où que nous soyons, ls ~/Downloads te permet de lister le contenu de ton dossier Downloads.

## Pour en savoir plus sur les commandes de base

<https://files.fosswire.com/2007/08/fwunixref.pdf>

# Terminal : Fichiers et dossiers

## Introduction

Le terminal est souvent utilisé pour manipuler des répertoires et des fichiers.

Avec ton terminal, tu peux créer et supprimer des répertoires, créer des fichiers, copier et déplacer des fichiers.

Dans cette quête, tu vas apprendre à utiliser toutes ces commandes dans ton terminal.

## Créer des répertoires

Commande **mkdir** → makedirs

Depuis la racine de ton dossier personnel (cd pour y retourner), entre:

```
mkdir quests
mkdir -p quests/shell/vegetables quests/shell/fruits
mkdir quests/shell/remove-me quests/shell/delete-me
ls quests/shell
```

Remarques :

- Tu peux créer un seul répertoire comme sur la première ligne, ou plusieurs à la fois comme sur la deuxième ligne.
- La commande de la deuxième ligne échouerait sans le flag -p parce que le répertoire quests/shell n'existe pas; -p indique à mkdir de créer les répertoires intermédiaires si ceux-ci n'existent pas.

## Supprimer des répertoires

Commande **rmdir** → removedirectory

Nous allons supprimer deux des répertoires que nous venons de créer:

```
rmdir quests/shell/remove-me quests/shell/delete-me
```

Attention, ici, rmdir ne fonctionne que sur des répertoires vides!

## Créer des fichiers

Commande **touch**

Cet exemple montre que tu peux créer un ou plusieurs fichiers, de la même manière que tu peux créer plusieurs répertoires avec mkdir.

```
cd ~/quests/shell
touch apple
touch apricot
touch carrot parsnip cauliflower courgette
ls
```

Créer des fichiers vides peut être utile de temps en temps, même si cela ne semble pas évident à première vue !

**touch** a un second effet : il modifie le timestamp du fichier. Un timestamp (horodatage) est une information précise de date/heure.

Le système de fichiers mémorise, entre autres, les horodatages de la dernière modification et du dernier accès au fichier.

A proprement parler, il ne s'agit pas d'un "historique", car seule la dernière date/heure est stockée. Par exemple :

```
ls -l ~/.bashrc
touch ~/.bashrc
ls -l ~/.bashrc
```

Le second 'ls -l' indique que le dernier horodatage de modification a été fixé à la date du jour.

## Copier des fichiers

Commande cp → copy

La commande cp est utilisée avec deux paramètres, à savoir la ou les source(s) et la destination (ou cible). Il peut y avoir plusieurs sources, mais il n'y a qu'une seule destination.

### Exemple 1 : Copier un fichier

Nous créons ici deux copies du fichier apple, qui est la source dans les deux cas, la destination étant à chaque fois un nouveau fichier : banana or orange, chacun étant une copie exacte de apple.

```
cp apple banana
cp apple fruits/orange
ls ; ls fruits
```

Remarques :

- Tu peux spécifier n'importe quel chemin pour la cible, comme indiqué dans le deuxième cp
- ; te permet de lancer plusieurs commandes en une seule ligne.

## Exemple 2 : Copier plusieurs fichiers dans un répertoire

Ici, nous créons une copie de chacun des fruits dans le répertoire fruits :

```
cp ap* banana fruits/  
ls fruits
```

Remarques :

- Lorsque l'ordre cp reçoit plus de deux arguments, il s'attend à ce que le dernier, la destination, soit un répertoire.
- Le / ajouté aux fruits est optionnel (même sans lui, cp déterminera si la cible est un fichier ou un répertoire).
- Le concept des caractères 'wildcard' : des caractères spéciaux permettent de spécifier simplement des ensembles de fichiers. Le caractère \* seul signifie "tous les fichiers" ; ap\* signifie "tous les fichiers commençant par ap" (donc "apple" et "apricot" correspondent).

## Exemple 3 : Faire une copie récursive

Tu rencontreras le terme récursivité si tu suis une formation de développeur.

Il s'agit de répéter la même opération un certain nombre de fois.

Appliqué à la copie, cela se traduit par copier un répertoire et son contenu entier.

Exemple avec le répertoire /fruits :

```
cp -R fruits copy-of-fruits
```

Concrètement, cp copie d'abord le répertoire donné comme argument source (ici fruits). Grâce à l'argument -R, si le répertoire source contient des fichiers, ceux-ci sont copiés. S'il contient des répertoires, il les copie aussi, puis "descend" dans chacun de ces répertoires, et copie leur contenu, et ainsi de suite.

Cela fonctionne également avec des répertoires sources multiples, et dans ce cas tu devras donner un répertoire existant comme argument de destination. Ici, nous allons copier fruits et copy-of-fruits dans le dossier /tmp qui contient les fichiers temporaires du système. Comme son contenu est effacé chaque fois que la machine est redémarrée, il peut être "pollué" sans crainte !

```
cp -R fruits copy-of-fruits /tmp/
```

Tu remarqueras que le flag -R est obligatoire pour copier un répertoire, même si il est vide.

Supprimer des fichiers et des répertoires

Commande **rm** → remove

Attention ! Contrairement au bouton "Mettre dans la corbeille" de Windows ou de MacOS, la suppression via rm est immédiate et définitive.



## Exemple 1 : Supprimer les doublons

Après les exemples de cp, tu te retrouves avec des doublons, que tu peux voir en lançant find - qui affiche le contenu de l'arborescence du répertoire courant.

Sachant que nous avons copié les fruits dans le répertoire fruits, nous allons supprimer les originaux, en utilisant rm :

```
rm ap* banana
```

Là encore, il est possible de passer plusieurs fichiers en paramètre, et d'utiliser une wildcard pour entrer moins de caractères.

## Exemple 2 : Supprimer récursivement un répertoire

La commande rmdir ne permettant pas d'effacer un répertoire non-vide, on peut utiliser rm avec l'option -r (récuratif) pour y parvenir :

```
rm -r copy-of-fruits
```

Comme le rm est immédiat et définitif, tu dois faire attention lorsque tu l'utilises avec le -r et/ou avec la wildcard \*.

Il est en effet trop facile de se tromper et d'effacer par inadvertance tout un répertoire et son contenu.

## Déplacer et/ou renommer

Commande **mv** → move

mv est utilisé avec deux ou plusieurs arguments, tout comme cp - le dernier étant la destination, et le(s) précédent(s) la (ou les) source(s).

- Déplacer : mv ca\* parsnip courgette vegetables/
- Renommer : mv fruits/orange fruits/grapefruit
- Déplacer et renommer : mv vegetables/courgette fruits/kiwi

## Pour aller plus loin

[https://linuxcommand.org/lc3\\_lts0050.php](https://linuxcommand.org/lc3_lts0050.php)

## Challenge

Tu vas devoir télécharger un fichier planets.zip, que tu récupéreras via le terminal.

Tu auras besoin de deux outils en ligne de commande : curl et unzip, qui te permettent respectivement de télécharger un fichier, et de décompresser une archive .zip.

Pour les installer, tu vas également utiliser le terminal ! Voici les commandes à exécuter dans une distribution « Debian like » (Debian, Ubuntu, Kali) :

```
sudo apt install -y curl unzip  
(sudo avant une commande t'accorde temporairement des privilèges administrateur,  
qui sont nécessaires pour installer un logiciel).
```

### Télécharger le fichier

Navigue maintenant dans le répertoire ~/quests/shell et lance ces deux commandes, pour télécharger l'archive planets.zip et la décompresser :

```
curl --ssl-no-revoke -L -o planets.zip  
"https://github.com/WildCodeSchool/quests-  
resources/blob/master/terminal/planets.zip?raw=true"  
unzip planets.zip
```

## Exercice

Cette archive contient des images de planètes réelles et fictives. Une fois que tu as extrait ses fichiers, utilise l'explorateur de fichiers de ton système pour visualiser les images qu'elle contient.

Mais le but n'est pas (encore) de devenir un touriste de l'espace !

Tu vas utiliser une partie des commandes que tu viens d'apprendre dans cette quête, pour classer les planètes. Tu vas devoir :

1.  Créer trois répertoires sous planets :
  - real
  - fictional
  - inhabited
2.  Créer trois autres répertoires dans le répertoire planets/real:
  - terrestrial
  - gas-giants
  - dwarf-planets
3.  Déplacer chacune des neuf planètes réelles à l'endroit correct parmi les trois sous-dossiers de planets/real,
4.  Déplacer les trois planètes fictives sous "planets/fictional",
5.  Copier les quatre planètes habitées (qu'elles soient habitées par des humains ou des robots) dans planets/inhabited (remarque : attention en anglais, inhabited signifie habité, uninhabited signifie inhabité)
6.  Enfin, puisque Pluton n'est plus vraiment considérée comme une planète, tu vas enfin l'effacer, ainsi que son répertoire parent.

Si tu es un peu perdu, tu peux demander aux camarades branchés science(-fiction) de t'aider, sinon tu peux trouver des indices [ici](#) et [ici](#).

Le résultat de tes efforts comprendra :

- Une copie de la sortie de la commande find (ou pour les utilisateurs de mac OS la sortie de la commande ls -R )
- Une copie de l'historique des commandes qui t'a conduit à ce point : utilise la commande history et colle uniquement les lignes relatives au challenge
- Après avoir validé ton résultat, tu peux nettoyer le répertoire ~/quests/shell.