

# TP Systèmes d'exploitation (rappels de 1ère)

## Objectifs :

- ▷ Utiliser les commandes de base en ligne de commande.
- ▷ Gérer les droits et permissions d'accès aux fichiers.

## 1 Introduction

Dans ce chapitre, nous allons revoir les commandes de base en ligne de commande, vues en classe de 1<sup>ère</sup>, notamment celles qui sont les plus utilisées dans les deux **systèmes d'exploitation** ou **OS** (*Operating System*) les plus courants : *Windows* et *Linux*.

Pour cela, au fur et à mesure que l'on présente les commandes systèmes, vous devez les tester sur chacun des systèmes d'exploitation.

## 2 Environnement de travail

### 2.1 Interpréteur de commandes

<i>Windows</i>	<i>Linux</i>
Interpréteur :	Interpréteur :
Les commandes sont tapées dans un terminal en mode texte.	Les commandes sont tapées dans un terminal en mode texte.
Pour démarrer l'interpréteur de commandes sous <i>Windows</i> , il vous suffit d'aller dans le menu <b>Démarrer</b> puis <b>cmd</b> .	Nous allons utiliser le site <i>Cocalc</i> qui simule une interface <i>Linux</i> .

### 2.2 Aide sur les commandes

#### 2.2.1 Liste des commandes

Un premier niveau d'aide vous permet d'avoir accès à la liste des commandes de base disponibles depuis votre interpréteur de commandes. Il vous suffit d'utiliser la commande **help**.

```
$ help
```

#### 2.2.2 Aide sur une commande particulière

Il est aussi possible d'obtenir de l'aide sur une commande particulière afin de connaître les options et les arguments pour l'utiliser.

```
$ help commande
```

Par exemple :

```
$ help cd
```

Retourne :

```
cd [-L|[-P [-e]] [-@]] [dir]
Change the shell working directory.

Change the current directory to DIR.  The default DIR is the value of the
HOME shell variable [...]
```

### 3 Répertoires

#### Rappel :

Imaginez une grande commode qui contient des tiroirs dans lesquels pourraient se trouver des fichiers et d'autres tiroirs,...,etc.

Un répertoire peut donc contenir :

- des fichiers,
- d'autres répertoires.

Sous <i>Windows</i>	Sous <i>Linux</i>
C: est la racine de votre disque dur D: est la racine de votre lecteur optique ( <i>BluRay</i> ) E: est la racine d'un lecteur réseau partagé, ..., etc.	/ est la racine unique sous <i>Linux</i> .  Il n'y a pas de lettre de lecteur car <i>Linux</i> ne donne pas de nom aux lecteurs comme le fait <i>Windows</i> . Il dit juste : « La base, c'est / ».

#### 3.1 Visualisation des répertoires

Sous <i>Windows</i>	Sous <i>Linux</i>
La commande <b>dir</b> ( <i>directory</i> ) permet de lister le contenu d'un répertoire.  Les options de ce tableau peuvent être utilisées séparément ou conjointement.  Par exemple : > <b>dir /S /P</b>	La commande <b>ls</b> ( <i>list subdirectory</i> ) permet de lister le contenu d'un répertoire.  Vous pouvez trouver les différentes options sur <a href="http://www.commandeslinux.fr/commande-ls/">http://www.commandeslinux.fr/commande-ls/</a> .  Par exemple : \$ <b>ls -l</b> affiche toutes les informations sur les fichiers et répertoires, du répertoire courant.
<div style="border: 1px solid black; padding: 5px; background-color: #f0f0f0;">             /A Affiche tous les fichiers y compris les fichiers caches              /N Affich le contenu du repertoire au format long              /P Affiche le contenu du repertoire en defilement par page              /S Affichage recursif du repertoire courant           </div>	
La commande <b>tree</b> permet l'affichage récursif du répertoire courant sous la forme d'un arbre. > <b>tree</b> Une des options importante est <b>F</b> pour aussi lister les fichiers.	La commande <b>tree</b> permet l'affichage récursif du répertoire courant sous la forme d'un arbre. \$ <b>tree</b>

#### 3.2 Comment se déplacer dans les répertoires ?

La commande **cd** (*change directory*) permet de se déplacer dans l'arborescence des fichiers. Elle s'utilise en lui donnant comme argument un répertoire.

```
$ cd repertoire
```

Il est possible d'utiliser des chemins relatifs et absolus. Ainsi, pour donner un chemin relatif on peut utiliser le **.** qui veut dire le répertoire courant ou **..** qui veut dire le répertoire parent.

Exemple pour remonter dans un répertoire parent :

```
$ cd ..
```

Sous <i>Windows</i>	Sous <i>Linux</i>
La commande <code>cd</code> , utilisée sans argument, vous informe sur le chemin du répertoire courant.	La commande <code>cd</code> , utilisée sans argument, vous informe sur le chemin du répertoire courant.
La commande <code>cd</code> utilisée avec pour argument <code>\</code> vous ramène à la racine de l'unité (partition) en cours. <code>&gt; cd \</code>	La commande <code>cd</code> , utilisée avec pour argument <code>/</code> , vous ramène à la racine. <code>\$ cd /</code>
Pour changer d'unité, il vous suffit de désigner par sa lettre la nouvelle unité sur laquelle vous souhaitez aller.	La commande <code>pwd</code> ( <i>print working directory</i> ) vous informe sur le chemin du répertoire courant.
Imaginons que vous possédez deux unités appelées <code>C:</code> et <code>D:</code> . Vous pouvez taper la lettre en majuscule ou en minuscule. <code>&gt; d:</code>	

### 3.3 Créer un répertoire

Sous <i>Windows</i>	Sous <i>Linux</i>
La commande <code>mkdir</code> ou <code>md</code> ( <i>make directory</i> ) permet de créer un nouveau répertoire (dossier).	La commande <code>mkdir</code> permet de créer un nouveau répertoire (dossier).
<code>\$ mkdir NouveauDossier</code> ou <code>\$ md NouveauDossier</code>	<code>\$ mkdir NouveauDossier</code>

Elle prend en argument le nom du répertoire que vous souhaitez créer.

Par exemple, la commande suivante :

```
$ mkdir dossier\dossier2
```

permet de créer un sous-répertoire `dossier2` à l'intérieur du répertoire `dossier`.

### 3.4 Copier un répertoire

Sous <i>Windows</i>	Sous <i>Linux</i>
Pour copier un répertoire et ce qu'il contient, il faut utiliser la commande <code>xcopy</code> qui permet de copier des fichiers ou répertoires et ce qu'ils contiennent.	La commande <code>cp -r</code> permet de copier un répertoire et ses sous-répertoires.
<code>\$ xcopy source destination /options</code>	<code>\$ cp -r source destination</code>
Options spécifiques à <code>xcopy</code> : <code>/S</code> : Copie les répertoires vides. <code>/E</code> : Copie les répertoires, y compris ceux vides. <code>/Y</code> : Confirme la suppression des fichiers existants sans demande de confirmation. <code>/H</code> : Copie également les fichiers cachés et systèmes.	Options spécifiques à <code>cp -r</code> : <code>-i</code> : Demande confirmation avant d'écraser les fichiers existants. <code>-u</code> : Ne copie que les fichiers plus récents que ceux déjà existants dans la destination. <code>-r</code> : Copie les répertoires de manière récursive.
Plus d'options disponibles.	Plus d'options disponibles.

### 3.5 Supprimer un répertoire

Sous <i>Windows</i>	Sous <i>Linux</i>
<p>La commande <b>rmdir</b> ou <b>rd</b> (<i>remove directory</i>) permet de supprimer un répertoire vide.</p> <p>Deux conditions permettent de supprimer un répertoire avec cette commande :</p> <ol style="list-style-type: none"> <li>1. On doit être à l'extérieur du répertoire.</li> <li>2. Il doit être vide.</li> </ol> <p>Par exemple, la commande suivante supprime le répertoire vide <b>test</b> qui a été créé dans <b>dossier</b>.</p> <pre>\$ rd dossier\test</pre>	<p>La commande <b>rmdir</b> permet de supprimer un répertoire vide.</p> <p>Répertoire vide :</p> <pre>rmdir test</pre>

## 4 Fichiers

### 4.1 Renommer un fichier

Sous <i>Windows</i>	Sous <i>Linux</i>
<p>La commande <b>rename</b> ou <b>ren</b> en abrégé, change le nom d'un fichier.</p> <p>Sa syntaxe générale est la suivante :</p> <pre>\$ rename ancien_nom nouveau_nom</pre>	<p>La commande <b>mv</b> (<i>move</i>) permet de déplacer ou de renommer des fichiers sous <i>Linux</i>.</p> <pre>\$ mv ancien_nom nouveau_nom</pre>

### 4.2 Supprimer un fichier

Sous <i>Windows</i>	Sous <i>Linux</i>
<p>La commande <b>del</b> ou <b>erase</b> supprime un ou plusieurs fichiers.</p> <p>Sa syntaxe générale est la suivante :</p> <pre>\$ del fichier1.txt fichier2.txt</pre> <p>Plusieurs options peuvent être utilisées avec cette commande. Par exemple :</p> <pre>\$ del /s monDossier\*.txt</pre> <p>va supprimer tous les fichiers textes dans <b>monDossier</b> ainsi que dans tous les sous-dossiers de <b>monDossier</b>.</p> <p>Remarquez le joker <b>*</b> qui permet des suppressions multiples en une seule opération.</p>	<p>La commande <b>rm</b> (<i>remove</i>) permet de supprimer un fichier ou un dossier sous <i>Linux</i>.</p> <pre>\$ rm fichier1.txt fichier2.txt</pre> <p>Plusieurs options peuvent être utilisées avec la commande <b>rm</b>.</p> <p>Vous pouvez consulter ce site <a href="http://www.commandeslinux.fr/commande-rm/">http://www.commandeslinux.fr/commande-rm/</a> pour plus d'informations.</p>

### 4.3 Copier un fichier

Sous <i>Windows</i>	Sous <i>Linux</i>
<p>La commande <code>copy</code> permet de copier un ou plusieurs fichiers d'une source à une destination.</p> <p>Si on met le nom de répertoire comme destination, le fichier copié a le même nom que le fichier source.</p> <pre>\$ copy test.txt dossier</pre> <p>Si on met un fichier en destination, alors le fichier copié aura le nom donné en destination.</p> <pre>\$ copy test.txt test2.txt</pre> <p>Afin de pouvoir copier plusieurs fichiers, il faut utiliser le caractère générique <code>*</code>. Il est possible d'utiliser deux caractères génériques : <code>*</code> signifie n'importe quelle combinaison de caractères et <code>?</code> signifie n'importe quel caractère.</p> <p>La commande suivante copie tous les fichiers ayant pour extension <code>.txt</code> à la racine de l'unité <code>d:</code>.</p> <pre>\$ copy *.txt d:</pre>	<p>La commande <code>cp</code> permet de copier un fichier ou plusieurs fichiers sous Linux.</p> <p>Si on met le nom de répertoire comme destination, le fichier copié a le même nom que le fichier source.</p> <pre>\$ cp test.txt dossier</pre> <p>En utilisant le caractère <code>+</code>, il est possible de copier le contenu de plusieurs fichiers dans un unique fichier. Par exemple, dans la commande suivante, les contenus de <code>test</code> et <code>test2</code> sont copiés dans le fichier <code>test3</code>.</p> <pre>\$ cp test.txt+test2.txt test3.txt</pre> <p>Vous pouvez consulter ce site <a href="http://www.commandeslinux.fr/commande-cp/">http://www.commandeslinux.fr/commande-cp/</a> pour plus d'informations sur la commande <code>cp</code>.</p>

### 4.4 Créer un fichier vide

Sous <i>Windows</i>	Sous <i>Linux</i>
<p>La commande <code>copy nul</code> permet de créer un fichier vide.</p> <pre>\$ copy nul test.txt</pre>	<p>La commande <code>touch</code> permet de créer un fichier vide.</p> <pre>\$ touch test.txt</pre>

## 5 Exercices

### Exercice 1 : *Linux cheatsheet*

A partir du [site suivant](#), faire un memento des commandes fondamentales *Linux* que vous devez connaître pour le Bac. Faire un tableau avec les commandes suivantes :

- `ls`
- `cd`
- `cp`
- `mv`
- `rm`
- `mkdir`
- `chmod`
- `chown`
- ainsi que les commandes de gestion de processus (`top`, `free`, `ps`, `kill`)

**Exercice 2 : Application directe**

Réaliser les actions suivantes sur *Windows* (cmd) et sur *Linux* sur le site *Cocalc* :

<https://cocalc.com/doc/linux.html>

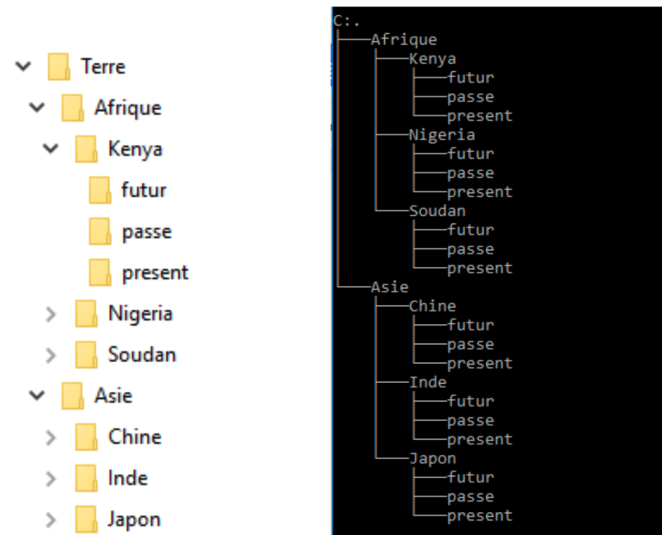
1. Créer un dossier **test** et un sous-dossier à l'intérieur de ce dossier **test2**.
2. Afficher l'arborescence de vos dossiers pour vérifier qu'ils ont bien été créés.
3. Dans le dossier **test2** créer un fichier vide **exemple.txt**.
4. Afficher l'arborescence de vos dossiers pour vérifier qu'ils ont bien été créés.
5. Créer une copie de **exemple.txt** et le nommer par **exemple2.txt**
6. Afficher l'arborescence de vos dossiers pour vérifier qu'il a bien été créé.
7. Créer une copie du dossier **test2** et le nommer **test3**.
8. Supprimer le dossier **test2**.
9. Est-ce que cela fonctionne ?

Pour les exercices 2 et 3, vous prendrez soin de compter le nombre d'opérations ou de commandes nécessaires pour réaliser chaque exercice.

**Exercice 3 : Mode graphique**

Admettons que vous soyez historien et que vous souhaitiez rédiger un texte concernant l'histoire, le présent et le futur de plusieurs pays. Vous devez donc constituer pour chaque pays un dossier qui doit contenir trois sous-dossiers qui contiendront les documents sur chacun des trois sujets principaux de l'histoire : **passee**, **present** et **futur**.

Un extrait de cette hiérarchie de dossiers se présentera de la façon suivante (ceci devrait vous servir de référence lors des exercices suivants) :

**Partie Afrique :**

1. Créer le répertoire **Terre** et à l'intérieur de celui-ci le répertoire **Afrique** et à l'intérieur de ce dernier le répertoire **Kenya**.
2. Sélectionner le répertoire **Kenya** et créez trois dossiers **passee**, **present** et **futur**.

La structure d'un pays est maintenant complète.

3. Dupliquer cette structure pour les autres pays en ajoutant les pays suivants : **Madagascar**, le **Cameroun** et l'**Afrique du sud** pour l'Afrique ainsi que **Singapour**, le **Vietnam** et le **Cambodge** pour l'Asie.

**Exercice 4 : Lignes de commandes sous Windows**

Faire les mêmes exercices mais avec l'interpréteur de commandes pour les continents **Asie**, **Europe**, **Amérique** et **Océanie**. N'oubliez pas d'aller chercher l'aide sur les commandes pour trouver les bonnes options qui peuvent diminuer le nombre de commande à réaliser pour faire une opération.

Vous devez utiliser les commandes **md** et **xcopy** !

**Exercice 5 : Lignes de commandes sous Linux**

Pour les TP nécessitant un terminal Linux, vous pouvez également utiliser ce site :

<http://s-macke.github.io/jor1k/demos/main.html>

À partir du répertoire courant et en utilisant les commandes **mkdir** et **touch**, créer l'arborescence suivante :

```
— dir1/ _____ fichier1
   |    | _____ .fichier2
   |
   | dir2/ _____ fichier3
```

1. Placez-vous dans le répertoire **dir1**. Lister les fichiers. Que constatez-vous ? Chercher l'option de la commande **ls** qui permet d'obtenir ce qui est attendu à l'aide de la commande **man ls**.
2. Donner le chemin relatif de **fichier3**.
3. Taper la commande **cd ..**. Donner à nouveau le chemin relatif de **fichier3**.
4. Taper la commande **cd dir2/** puis la commande **cp fichier3 ../dir2/fichier4**. Vérifier que le **fichier4** a bien été créé.
5. Taper la commande **cd ..** puis la commande **rmdir dir1**. Que se passe-t-il ? Expliquer pourquoi à l'aide de la commande **man rmdir**.
6. Déplacer l'ensemble des fichiers contenus dans le répertoire **dir1** dans le répertoire **dir2** en saisissant depuis le répertoire courant ; **mv dir1/\* dir2/**  
\* est un métacaractère (*wildcard* en anglais), il remplace tout les nom de fichiers.  
**\*.jpg** signifierait tous les fichiers dont l'extension est **jpg**.
7. Effacer le répertoire **dir1/**

**Exercice 6 : Redirection dans un fichier**

1. Dans le répertoire courant, taper : **echo 'Longtemps, je me suis couché de bonne heure.'**
2. Dans le répertoire courant, taper : **echo 'Longtemps, je me suis couché de bonne heure.' > debut.txt**
3. Lire le fichier **debut.txt** à l'aide de la commande **cat debut.txt**
4. Taper ensuite : **echo 'Parfois, à peine ma bougie éteinte, mes yeux se fermaient si vite que je n'avais pas le temps de me dire : « Je m'endors. » ' > debut.txt**
5. Lire le fichier **debut.txt**.

## Pour mieux comprendre :

- ▷ `commande > fichier` envoie le contenu de la sortie standard ( par défaut le terminal) de la commande dans le fichier `fichier` en écriture avec écrasement du contenu précédent.
- ▷ `commande » fichier` envoie le contenu de la sortie standard de la commande dans le fichier `fichier` en écriture avec ajout à la fin du fichier.
- ▷ `echo texte` envoie `texte` dans la sortie standard
- ▷ `cat NomFichier` affiche le contenu de `NomFichier`

**Exercice 7 :** *D'autres possibilités du shell*

Taper chacune des lignes de commandes et observez le résultat.

1. `curl wttr.in/paris`
2. `curl wttr.in/moon`
3. `cal`
4. `date`
5. `echo "4+6" | bc -l`
6. `echo "scale=10; 4*a(1)" | bc -l`
7. `ls | sort -r` ( à comparer avec `ls` seul)
8. `echo 'Un palindrome' | rev`
9. `echo $((RANDOM %6 +1))` (la répéter plusieurs fois)
10. `yes` (saisir `Ctrl + C` pour arrêter)
11. `firefox ifs.edu.sg &`
12. `factor 12` puis `factor 13` puis `factor 64`
13. `while true; do echo "$(date '+%D %T')"; sleep 1; done` (`Ctrl + C` pour arrêter)

**Exercice 8 :** *Script Shell sous Linux*

Dans cet exercice nous allons créer un script *Shell*. Pour cela :

1. Aller sur [Cocalc](#).
2. Aller dans l'onglet *New* à côté de *Files* en haut de la page.
3. Entrer le nom du fichier, par exemple `script`.
4. Dans la liste déroulante *More file types*, choisissez *Shell(.sh)*.
5. Cela crée et ouvre le nouveau fichier dans un éditeur texte.
6. Appuyer sur le bouton `>_Shell` dans la barre d'outils. Cela ouvre un terminal à droite de l'écran.
7. Ne pas oublier d'utiliser le bouton de sauvegarde avant de tester votre script.
8. Recopier le programme ci-dessous dans le fichier `script.sh`

```

1  #!/bin/bash
2  for i in $(seq 1 10)
3  do
4      if test -d TP$i
5      then
6          echo "TP$i existe deja"
7      else
8          echo "creation de TP$i"
9          mkdir "TP$i"
10     fi
11 done

```



9. Exécuter le programme en effectuant la commande : `./script.sh`
10. Que se passe-t-il ?
11. Pour régler le problème, faire des recherches sur la commande `chmod`.
12. Commenter ce script en décrivant précisément ce qui se passe à chaque ligne. Comme en Python, les commentaires en shell commencent par un `#` et finissent à la fin de la ligne.
13. Améliorer le programme pour qu'il puisse créer dans chaque dossier seulement s'il existe déjà, 5 fichiers nommés `new_file_1.txt`, `new_file_2.txt`, ..., `new_file_5.txt`. Vous pouvez utiliser la commande `touch "TP$i/new_file_$j.txt"`
14. Pouvons-nous trouver une commande pour supprimer en une seule fois tous les dossiers et fichiers que nous venons de créer ?

### Exercice 9 : *Terminus*

Dans cet exercice vous allez découvrir un certain nombre de commandes *Linux/UNIX*, utilisable dans le *shell* à travers un escape game en ligne : *Terminus*.

Vous trouverez ce jeu à l'url suivante : <http://luffah.xyz/bidules/Terminus/>

Jouer à *Terminus* et compléter au fil du jeu :

- un tableau des commandes découvertes avec leur fonctionnalité et leur syntaxe
- une représentation du monde de *Terminus* sous la forme d'une arborescence, comme celle d'un système de fichiers.

On donne ci-dessous un tableau des commandes disponibles au début du jeu :

Commande	Description	Syntaxe
<code>cat</code>	Afficher dans la console	Saisir <code>cat</code> <code>Objet</code> ou <code>cat</code> <code>Personne</code>
<code>ls</code>	Lister les éléments	Saisir <code>ls</code>
<code>cd</code>	Changer de destination	<code>cd ..</code> pour revenir à l'emplacement précédent
<code>cd</code>	Changer de destination	<code>cd Salle</code> pour entrer dans Salle
<code>cd</code>	Changer de destination	<code>cd ~</code> pour revenir au point de départ

### Exercice 10 : *Find your path*

1. Aller sur le site <https://demo710.univ-lyon1.fr/FYP/>
2. Appuyer sur F1 pour basculer en français.
3. Indiquer les chemins relatifs ou absolus pour aller vers le dossier indiqué en vert à partir du dossier courant (en rose).

Cela peut paraître étonnant pour des personnes n'ayant rien connu d'autre que l'interface graphique utilisateur, mais le *Terminal* (aussi appelé *console*) est un outil de travail indispensable pour de nombreux développeurs et Ops. Il y a plusieurs raisons à cela :

Le Terminal (aussi dit "Console" ou "Interface de ligne de commande") permet de donner des instructions à l'ordinateur.

Comme ce n'était pas simple, les Interfaces Utilisateur sont apparues fin 1970's pour rendre les ordinateurs plus accessibles...

...Donc pourquoi t'obstiner à tout faire en ligne de commande... ?!

C'est plus pratique ! ... ben voyons...

Hmm... Quelle est la commande... ? Fichtre ! J'y comprends pas ! C'est parti mon kiki !

Déjà, je n'ai souvent pas le choix lorsque je travaille sur un serveur distant.

J'envoie directement mes commandes au système d'exploitation (OS) via un interpréteur de commandes.

Il y a rarement une interface, je dois écrire mes commandes ! \*

\* ou les automatiser, c'est encore mieux !

OS

- Linux
- MacOS
- Windows
- ...

Interpréteur de commandes

- sh
- bash
- zsh
- ...

Ouvre le fichier

SCHEUR APPLICATIF

Contrairement à l'interface, les commandes varient très peu d'un OS ou d'un interpréteur à l'autre.

Cela facilite donc le partage d'informations : c'est plus pratique de recopier une commande que d'expliquer où cliquer.

Donc si tu cherches une commande, Internet est ton ami ! Quelqu'un se sera toujours posé la question avant toi... !

De rien.

Commande : renommer ses photos

Ouvre le dossier, c'est bon ? Yes! Après ?

Ubuntu

- sh
- cd
- grep
- apt install

MacOS

- sh
- cd
- grep
- brew install

Et enfin, en cas d'erreur, j'ai un message d'erreur détaillé.

Parfois même un pouton...

Après, chacun son terminal et chacun ses habitudes : shell, couleurs, raccourcis...

... jusqu'à obtenir ton terminal rêvé ! Elle est folle...

Et toi ? Tu utilises quelles commandes ? Hehe ! Il existe même une commande pour répondre à cette question ! Tadam !

GESTION DE FICHIERS

- cd : changer de dossier
- chmod : changer les droits d'accès à un fichier ou dossier
- cp : copier un fichier ou dossier
- grep : rechercher une expression dans un fichier ou dossier
- gzip : compresser un fichier en .gz
- ls : lister les fichiers ou dossiers présents
- open : ouvrir un fichier
- rm : supprimer un fichier ou dossier

RÉSEAU

- dig : obtenir l'IP d'un domaine
- ping : vérifier la disponibilité d'un serveur
- ssh : se connecter à un serveur distant
- topdump : écouter le trafic sur une interface réseau
- wget : télécharger un fichier

GESTION DES PROCESSUS

- kill : envoyer un signal à un processus
- ps : afficher les processus actifs

DONNÉES SYSTÈME

- date : afficher la date et l'heure
- du : afficher l'espace dossier utilisé
- which : localiser une commande

COMMANDES OUTIL

- { brew, apt, yum, ... } install <tool>
- <tool> : lancer une commande liée à l'outil

Et pour plus d'infos sur une commande, tapez "man" ou "tldr" + <commande> dans le terminal !

Je suis