Chapitre 16 - Systèmes d'exploitation

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 $1^{ère}$, 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

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

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

3.1 Visualisation des répertoires

Sous *Windows*	Sous *Linux*

La commande `dir (directory)` permet de lister le contenu d'un répertoire.
 	
Les options de ce tableau peuvent être utilisées séparément ou conjointement.
 	
Par exemple :
`> dir /S /P`
 	
`/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`
 	
La commande `tree` permet l'affichage récursif du répertoire courant sous la forme d'un arbre.
`> tree`
Une des options importante est `F` pour aussi lister les fichiers.
	La commande `ls (list subdirectory)` permet de lister le contenu d'un répertoire.
	
	Vous pouvez trouver les différentes options sur http://www.commandeslinux.fr/commande-ls/.
	
	Par exemple :
	`$ ls -l` affiche toutes les informations sur les fichiers et répertoires, du répertoire courant.
	
	La commande `tree` permet l'affichage récursif du répertoire courant sous la forme d'un arbre.
	`$ tree`

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 *Windows*	**Sous *Linux***

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

3.3 Créer un répertoire

Sous *Windows*	**Sous *Linux***
La commande `mkdir` ou `md` (*make directory*) permet de créer un nouveau répertoire (dossier). `$ mkdir NouveauDossier` ou `$ md NouveauDossier`	La commande `mkdir` permet de créer un nouveau répertoire (dossier). `$ mkdir NouveauDossier`

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

3.5 Supprimer un répertoire

Sous *Windows*	**Sous *Linux***
La commande `rmdir` ou `rd` (*remove directory*) permet de supprimer un répertoire vide. Deux conditions permettent de supprimer un répertoire avec cette commande : 1. On doit être à l'extérieur du répertoire. 2. Il doit être vide. Par exemple, la commande suivante supprime le répertoire vide `test` qui a été créé dans `dossier`. `$ rd dossier\test`	La commande `rmdir` permet de supprimer un répertoire vide. Répertoire vide : `rmdir test`

4 Fichiers

4.1 Renommer un fichier

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

4.2 Supprimer un fichier

Sous *Windows*	**Sous *Linux***

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

4.3 Copier un fichier

Sous *Windows*	**Sous *Linux***

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

4.4 Créer un fichier vide

Sous *Windows*	**Sous *Linux***
La commande `copy nul` permet de créer un fichier vide. `$ copy nul test.txt`	La commande `touch` permet de créer un fichier vide. `$ touch test.txt`

5 Exercices

Exercice 1 : *Linux cheatsheet*

A partir du site suivant, faire un mémento des commandes fondamenales *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

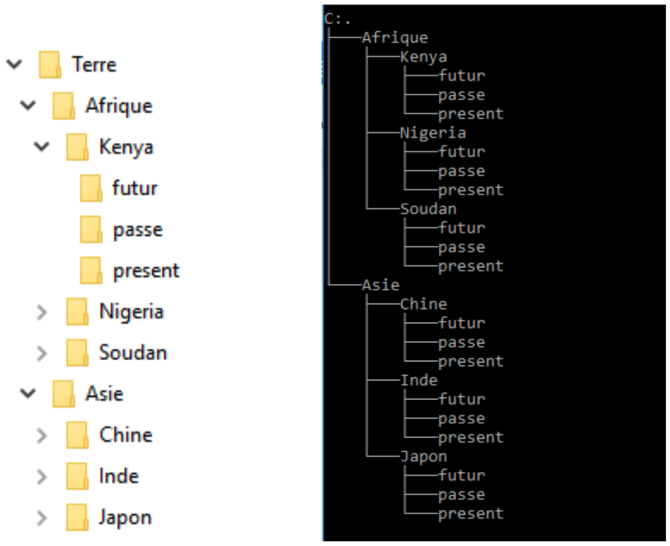
1. Créer un dossier `test` et un sous-dossier à l'intérieur de ce dossier `test2`.
2. Afficher l'arborescence de vos dossiers pour vérifier qu'ils ont bien été créés.
3. Dans le dossier `test2` créer un fichier vide `exemple.txt`.
4. Afficher l'arborescence de vos dossiers pour vérifier qu'ils ont bien 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 : `passe`, `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 `passe`, `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.

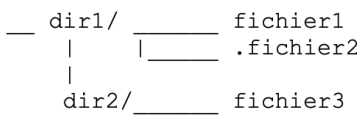
Vous devez utiliser les commandes `md` et `xcopy` !

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 :



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. Observer les résultats obtenus et faire des recherches sur Internet pour comprendre ce que font chacune des commandes.

1. `curl wttr.in/singapore` (Fonctionne également dans le terminal *Windows* si ne marche pas sur *Cocalc*)
2. `cal`
3. `date`
4. `echo "4+6" | bc -l`
5. `echo "scale=10; 4*a(1)" | bc -l`
6. `ls | sort -r` (à comparer avec `ls` seul)
7. `echo 'Un palindrome' | rev`
8. `echo $(($RANDOM %6 +1))` (la répéter plusieurs fois)
9. `yes` (saisir *Ctrl + C* pour arrêter)
10. `factor 12` puis `factor 13` puis `factor 64`
11. `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`

```
#!/bin/bash
for i in $(seq 1 10)
do
    if test -d TP$i
    then
        echo "TP$i existe deja"
    else
        echo "creation de TP$i"
        mkdir "TP$i"
    fi
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 denommés `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 nombres 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**
`cat`	Afficher dans la console	Saisir `cat Objet` ou `cat Personne`
`ls`	Lister les éléments	Saisir `ls`
`cd`	Changer de destination	`cd ..` pour revenir à l'emplacement précédent
`cd`	Changer de destination	`cd Salle` pour entrer dans Salle
`cd`	Changer de destination	`cd ~` 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 :