TP 02 - Le shell -

1 Le shell, c'est quoi ?

C'est une couche logicielle qui fournit l'interface utilisateur d'un système d'exploitation. Il correspond à la couche la plus externe de ce dernier. Deux méthodes d'accès au Shell sont possibles:

- Le mode console qui affiche un shell unique en plein écran, c'est l'interface homme machine de base du système d'exploitation;
- Le mode terminal qui émule[1] une console et qui affiche en général le shell dans une fenêtre sur l'écran.

Sous GNU/Linux les consoles Shell sont au nombre de six par défaut. Ces consoles sont accessibles depuis l'interface graphique avec les raccourcis Ctrl + Alt + Fx (Fx touches de fonction F1, F2, . . . , F6).

Il existe de nombreux shells qui se classent en deux grandes familles:

- La famille *C shell* (**Ex.** csh, tcsh);
- La famille *Bourne shell* (**Ex.** sh, bash, ksh).

Sous le système GNU/Linux, le shell par défaut est *bash*[2].

2 Les répertoires importants

- / la racine du système, il contient tous les autres dossiers et fichiers;
- **/bin** contient des programmes utilisés par tous les utilisateurs;
- **/sbin** contient des programmes système importants;
- **/boot** fichiers permettant le démarrage du système;
- **/usr** c'est dans ce dossier qui vont s'installer la plupart des programmes demandés par l'utilisateur;
- **/home** répertoires personnels des différentes personnes ayant un compte sur l'ordinateur;
- **/etc** répertoire contenant les fichiers de configuration du système;
- **/root** répertoire personnel du superutilisateur;
- **/lib** il contient les bibliothèques partagées (des fichiers .so l'équivalent des .dll sous Windows);

[1] Chercher à imiter
[2] Bourne Again SHell

- **/media** lorsqu'un périphérique amovible est inséré, vous pouvez y accéder à partir d'un sous dossier de media;
- **/tmp** répertoire temporaire dans lequel les programmes peuvent stocker des fichiers;
- **/var** contient des données variables, souvent des *logs*[3].

Répertoires particuliers

- **.** répertoire courant;
- **..** répertoire parent;
- **~** répertoire maison (home) de l'utilisateur.

Fichiers cachés Sous Linux et Unix, les fichiers cachés commencent par un point.

3 Le prompt

Pour signifier que le shell est prêt à recevoir des commandes, il affiche un *prompt*. Ce prompt peut contenir un nombre variable d'informations selon la configuration.

Exemple

```
etudiant@etudiant-pc:~$
```

- **etudiant** utilisateur courant;
- **etudiant-pc** nom de la machine;
- **~** répertoire courant;
- **$** type de l'utilisateur ($ dans le cas d'un utilisateur normal et # dans le cas du super-utilisateur (root)).

4 Syntaxe d'une commande

La syntaxe générale d'une commande est la suivante:

```
$ nom_commande [option...] [argument...]
```

5 Quelques commandes utiles

5.1 ls

```
$ ls [-alF] chemin_1 chemin_2 ... chemin_n
```

chemin_i est un nom de fichier ou de répertoire.

- **-l** Affiche plus d'informations (taille, date, droits, ...);
- **-a** Liste tous les fichiers y compris les fichiers cachés;
- **-F** Indique le type de fichier (ajoute * pour un exécutable, / si c'est un répertoire).

[3]Traces écrites de ce qui s'est passé récemment sur l'ordinateur

5.2 echo

```
$ echo [-n] message
```

Affiche un message. L'option **-n** supprime le saut de ligne.

5.3 cat

```
$ cat [-n] [fichier1 ...]
```

Recopie les fichiers spécifiés l'un après l'autre sur la sortie standard. Si aucun fichier n'est spécifié, la commande lit sur l'entrée standard jusqu'à rencontrer un caractère de fin de fichier Ctrl + D. L'option **-n** numérote les lignes.

5.4 cd

```
$ cd [chemin]
```

Change le répertoire courant. Sans argument, la commande ramène l'utilisateur dans son répertoire connexion (home).

5.5 cp

```
$ cp [-ir] source... dest
```

Copie le ou les fichier(s) *source* vers *dest*.

- **-i** Demander confirmation en cas d'écrasement de la destination;
- **-r** Copie récursive.

5.6 pwd

```
$ pwd
```

Affiche le répertoire courant.

5.7 mkdir

```
$ mkdir [chemin]
```

Crée un répertoire. Le chemin peut être relatif ou absolu.

5.8 mv

```
$ mv [-i] source dest
```

Si *dest* est un répertoire, déplace le fichier *source* vers *dest*. Si *dest* est un fichier, renomme *source*. L'option **-i** permet de demander confirmation en cas d'écrasement de la destination.

5.9 rm

```
$ rm [-ri] fichier ...
```

Supprime le ou les fichiers spécifiés.

- **-i** Demander confirmation pour chacun;
- **-r** Agit de façon récursive. Détruit aussi les répertoires et leurs sous-répertoires.

5.10 Autres commandes

`touch`, `ln`, `less/more`, `file`, `gzip/gnuzip`, `wc`, `grep`, `diff`, `chmod`, `find`, `ps`, `kill`, `fg`, `bg`, `ssh`, `passwd`, `man`, ...

6 Exercice

1. Lancer un terminal;
2. Taper chacune des commandes suivantes, observer le résultat:

```
$ pwd
$ ls
$ ls -F
$ ls -l
$ ls -lF
$ ls -a
$ ls -laF
$ clear
$ echo Hello wolrd
$ echo -n Hello world
$ touch def_prog.txt
$ ls -l
```

3. Éditer le fichier *def_prog.txt* avec l'éditeur **nano** en utilisant la commande suivante:

```
$ nano def_prog.txt
```

4. Saisir le texte suivant: "*L'art de programmer, c'est l'art de faire résoudre des problèmes par des machines. Il s'agit bien d'un art, au sens de l'artisan, qui passe par une longue période d'apprentissage et d'imitation. Dans cet exercice certains individus ont des dispositions naturelles; pour les autres un apprentissage rigoureux fournit les rudiments d'une méthode. Le reste est affaire de travail.*";
5. Sauvegarder votre fichier (Ctrl + O);
6. Quitter l'éditeur de texte (Ctrl + X).

```
$ wc def_prog.txt
$ cat def_prog.txt
$ cat -n def_prog.txt
$ cd Documents
$ clear
$ pwd
$ cp ../def_prog.txt .
$ ls
$ cd ..
$ pwd
$ rm def_prog.txt
$ mv Documents/def_prog.txt .
$ ls -l
$ ls -l Documents
$ mkdir prog
$ mkdir prog/c
$ cd prog
$ ls
```

7. Autres commandes:

```
$ cal
$ cal 2015
$ cal 5 2015
$ date
$ date +"%d/%m/%y - %H:%M"
$ date -d"31 dec 2014" +"%A"
```