

8.2

Découvrir les lignes de commandes

NSI 1ÈRE - JB DUTHOIT

8.2.1 Emulateur

Afin d'éviter toute installation, tout en disposant toujours du même environnement de base, disponible depuis toute machine connectée à Internet, vous travaillerez sur un émulateur linux <https://bellard.org/jslinux/vm.html?url=https://bellard.org/jslinux/buildroot-x86.cfg>

```
Loading...  
  
Welcome to JS/Linux (x86)  
  
Use 'vflogin username' to connect to your account.  
You can create a new account at https://vfsync.org/signup .  
Use 'export_file filename' to export a file to your computer.  
Imported files are written to the home directory.  
  
[root@localhost ~]# █
```

Remarque

L'utilisateur est **root** et le nom de la machine est **localhost**

👉 Essayer les commandes **whoami** et **hostname**

Remarque

L'utilisateur **root** est un **super administrateur** : c'est un utilisateur qui possède tous les droits.

8.2.2 Créer des utilisateurs

Il est possible d'ajouter un utilisateur **adduser toto**.

Pour se connecter avec tant que toto, taper **su toto!** Vous pouvez observer que le nom d'utilisateur est bien changé :

```
[root@localhost ~]# su toto  
[toto@localhost root]$ █
```

Pour se reconnecter en tant que super administrateur, taper **su**

Ajouter également un second utilisateur nommé **tata** ;

Définition

Une façon d'interagir avec le système est d'utiliser un terminal (ou console) dans lequel vous pourrez taper des commandes (qui ne sont rien d'autre que des programmes). Le programme avec lequel vous interagissez pour exécuter les commandes s'appelle le **shell**.

8.2.3 Aide sur les commandes

Le plus important à retenir, surtout quand on connaît une commande mais pas ses options, ou si l'on découvre une commande inconnue dans un exemple, est de savoir accéder à l'**information intégrée** au shell :

- **man la_commande** renvoie une aide complète (souvent plus longue qu'une page, on peut ajouter à la suite **less** pour se déplacer dedans avec les flèches, puis quitter avec **q**, voir à la suite pourquoi). **info la_commande** offre un service comparable.
- Plus simplement, la plupart des commandes ont une option **-help** (en général, double tiret pour une option qui s'écrit sur plus d'un caractère) ou **-h** qui décrit leur utilisation.
☞ Avec **ls -help**, trouver comment lister le contenu d'un répertoire en triant les éléments par ordre décroissant de taille.

8.2.4 Explorer l'arborescence

Utiliser les commandes **shell** qui permettent de se déplacer dans l'arborescence, de visualiser les répertoires, fichiers, ainsi que des informations sur eux :

1. Exécuter **pwd** qui donne l'adresse du répertoire dans lequel on se trouve. La noter. Est-ce une **adresse absolue** ou une **adresse relative**? Comment le voit-on ?

A RETENIR :

- / : Répertoire racine
- ~ : Répertoire d'accueil (home)
- . : Répertoire courant
- .. : Répertoire parent

2. Pour lister le contenu du répertoire courant, utiliser **ls**. Combien d'éléments contient-il ?
3. On peut ajouter des options à la commande :

- **ls -a** permet de lister également les fichiers cachés, à savoir ceux qui ont un nom commençant par **.** et qui sont invisibles par défaut.
On voit apparaître notamment le **.** qui représente le répertoire courant, ainsi que **..** qui désigne le dossier parent situé au niveau juste au dessus dans l'arborescence.
- **ls -l** donne une description plus complète.
Le premier caractère de chaque ligne indique s'il s'agit d'un fichier classique **-** ou d'un répertoire (**d** pour directory). On peut trouver également l'utilisateur propriétaire (celui qui l'a créé, à moins qu'on ne l'ait changé ensuite) et son groupe, la taille du fichier, sa date de dernière modification...
- On peut combiner les options : **ls -a -l** ou **ls -al**. Donner le nom d'un répertoire, celui d'un fichier ordinaire caché et celui d'un fichier ordinaire non caché.
- On peut aussi donner en **argument** le dossier à lister.
☞ Exécuter **ls /** ou **ls -l /** et indiquer le nombre de répertoires immédiatement présents à la racine de l'arborescence (Ignorer le linuxrc qui est un lien symbolique comme l'indique le 1 en début de ligne...)
☞ Vérifier avec **pwd** que vous êtes toujours dans le même dossier.

4. **cd** permet de se déplacer dans les répertoires. On lui donne en argument (écrit après un espace) l'adresse relative ou absolue où l'on veut se rendre.

- **cd /home** permet ainsi de se placer dans le sous-répertoire **home**. On peut commencer à écrire le nom du dossier puis utiliser l'**autocomplete** en appuyant sur la touche de tabulation).
- ☞ Combien de dossiers contient ce répertoire ? Faire le lien avec la création des utilisateurs. Ajouter un troisième utilisateur et vérifier votre conjecture.
- On peut toujours revenir à la base de son espace personnel avec **cd**

5. À l'aide des commandes précédentes, représenter l'arborescence (partielle) des fichiers en partant de la racine **/**. Il est possible de l'écrire sous la forme suivante :

```

/
|--- bin
|--- dev
|--- home
    |--- toto

```

Se limiter à un niveau de profondeur, sauf pour le dossier **home**

8.2.5 Créer, copier, déplacer, supprimer

Créer un dossier, créer un fichier

Se connecter en tant qu'utilisateur **toto**.

1. Créer dans le répertoire **toto** le dossier **interros** en utilisant la commande **mkdir**
2. Créer dans le répertoire **interro** le fichier **interro1.txt** en utilisant la commande **touch**.

Écrire dans une fichier, et lire un fichier

De façon générale, il est possible d'utiliser la commande **echo** pour écrire dans un fichier.

1. Se placer dans le répertoire **interro** et entrer **echo "exercice 1:">interro1.txt**
 - ☞ Lire le fichier avec la commande **cat** et observer le résultat.
2. Entrer **echo "exercice 2:">interro1.txt**
 - ☞ Lire le fichier avec la commande **cat** et observer le résultat.
3. Afin de ne pas écraser le fichier, il est possible d'écrire : **echo "exercice 3:">>interro1.txt**
 - ☞ Lire le fichier avec la commande **cat** et observer le résultat.
4. Tester **echo "exercice 1:">interro2.txt**
 - ☞ Que se passe-t-il ?

Déplacer, copier

1. Déplacer ce fichier à la base de ton espace personnel :

`mv interro1.txt ~`

Le `~` final peut être remplacé par `.` si on est déjà situé au bon endroit.

☞ S'entraîner à déplacer les fichiers d'un endroit à un autre, en utilisant soit des références absolues, soit des références relatives.

2. Créer un nouveau dossier local `exo` `mkdir exo` (make directory). Copier le fichier `interro1.txt` dans ce dossier avec la commande `cp` qui fonctionne comme `mv` mais conserve le fichier d'origine.

Supprimer un fichier

Détruire `interro2.txt` avec `rm interro2.txt` (remove).

Exercice 8.1

On considère l'arborescence suivante :

```
/  
|---r  
|   |---a  
|   |   |---d  
|   |   |   |f3  
|   |   |   |f4  
|   |  
|   |---b  
|   |   |f2  
|   |   |---e  
|   |   |   |f5  
|   |  
|   |f1  
|   |---c  
|   |   |---f  
|   |   |   |f6  
|   |   |---g  
|   |   |   |f7  
|   |   |   |f8
```

1. On se place dans le dossier `b`, et on utilisera ici des **références relatives**.
 - Quel est le résultat de la commande `ls` ?
 - Quelle commande faut-il utiliser pour lister le contenu du répertoire `a` ?
 - Quelle commande faut-il exécuter pour que la commande `ls` affiche `f7 f8` ?
2. On se place dans le répertoire `b`, et on utilisera ici des **références relatives**.
 - Quelles commandes faut-il exécuter pour copier le fichier `f1` dans le répertoire `d` et afficher le contenu de ce répertoire ?
 - Quelle(s) commande(s) faut-il ensuite effectuer pour que la commande `pwd` affiche `/r/c/f` ?
3. Dans l'arbre de fichier (peu importe l'endroit, donc on utilisera des **références absolues**), quelles commandes faut-il exécuter pour :
 - Créer un répertoire `h` dont le répertoire parent est `b`.
 - Déplacer les fichiers qui sont dans le répertoire `c` (éventuellement ses sous-répertoires) dans le nouveau répertoire `h`.
 - Dessiner l'arborescence après ces commandes

4. Dans l'arbre de fichier (peu importe l'endroit, donc on utilisera des **références absolues**), quelles commandes faut-il exécuter pour supprimer la branche **a** de l'arbre des fichiers ?
 5. En utilisant un émulateur linux :
 - a) Construire dans le répertoire / le répertoire **r** et ses sous-répertoires.
 - b) Vérifier les résultats aux questions précédentes.
- ***

Exercice 8.2

Une suite de commande permet à l'utilisateur d'automatiser certaines tâches. On parle alors de **script**, stocké dans un fichier d'extension **.sh**.

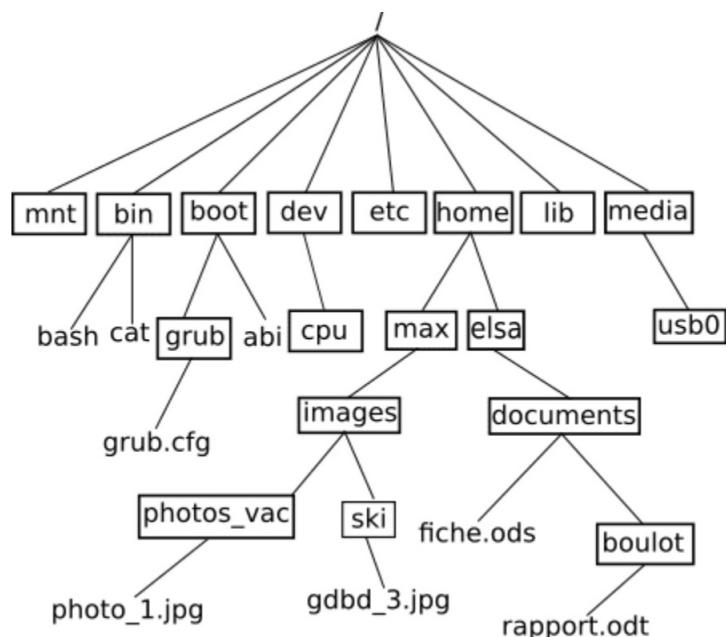
1. Se placer dans le répertoire racine /
2. Créer un fichier **auto.sh**
3. Ajouter au fichier les lignes suivantes :

```
mkdir ./nv_dossier
touch ./nv_dossier/nv_fichier.txt
echo "Hello World !" > ./nv_dossier/nv_fichier.txt
```

4. exécuter le fichier **auto.sh** en tapant **bash auto.sh** et vérifier le résultat.

Exercice 8.3

On considère l'arborescence suivante :



1. On se trouve dans le dossier **elsa** et on souhaite se déplacer en une seule commande dans le dossier **boulot** :
 - a) Donner la commande avec des références absolues
 - b) Donner la commande avec des références relatives
2. On se trouve toujours dans le dossier **elsa** et on souhaite ici se déplacer en une seule commande dans le dossier **grub** :
 - a) Donner la commande avec des références absolues

- b) Donner la commande avec des références relatives (celle qui ne commence pas par `./`)
3. On se trouve maintenant dans le dossier `photos_vac` et on souhaite avoir la liste non détaillée des fichiers et dossiers se trouvant dans le dossier `boulot`
- Donner la commande avec des références absolues
 - Donner la commande avec des références relatives (celle qui ne commence pas par `./`)