# Console Linux

~

La console, sous Linux, est l'interpréteur de commandes. Il s'agit d'une application faisant partie des composants de base du système d'exploitation. Sa fonction est d'interpréter les commandes qu'un utilisateur tape au clavier dans l'interface en ligne de commande. Généralement, sous Linux cette application est **bash**. Elle permet d'exécuter des commandes de base, mais il s'agit également d'un langage de script permettant d'exécuter des tâches complexes.
Nous nous contenterons ici de voir les commandes de base, ainsi que la gestion des droits et permissions d'accès aux fichiers.
Pour accéder à la console, il suffit généralement de lancer le programme Terminal dont l'icône est une fenêtre noire.

## 1. Commandes de base

**Exercice 1**
Tester au fur et à mesure les commandes et exemples ci-dessous :

1. Pour savoir dans quel répertoire de l'arborescence de fichiers on se situe, autrement dit le répertoire courant, on utilise la commande **pwd** (Print Working Directory).

   Généralement en démarrant la console on se situe dans le répertoire personnel, à l'adresse `/home/nomutilisateur`.

2. Pour changer de répertoire, on utilise la commande **cd** (Change Directory).
   - Utilisée sans argument (commande « `cd` »), elle permet de retourner à son répertoire personnel. On peut aussi utiliser la commande « `cd ~` » pour cela.
   - La commande « `cd ..` » permet de remonter d'un niveau dans l'arborescence.

   On peut utiliser le **chemin absolu**, autrement dit taper l'adresse complète du répertoire à partir de la racine (/), par exemple « `cd /home/nomutilisateur/Documents` ».

   On peut aussi utiliser le **chemin relatif**, c'est à dire une adresse déterminée à partir du répertoire courant. Pour cela on peut utiliser les « `..` » permettant de remonter d'un répertoire. Par exemple, « `cd ../Téléchargements` » permet d'aller dans le répertoire `/home/nomutilisateur/Téléchargements` si on était dans le répertoire :

   `/home/nomutilisateur/Documents`.

3. Pour avoir un manuel d'utilisation d'une commande donnée, on utilise la commande **man**.

   Par exemple, « `man pwd` ».

4. La commande **ls** (LiSt) permet de lister les fichiers et répertoires du répertoire courant. Elle possède de nombreuses options (à préciser après un tiret si on en utilise). Pour afficher tous les fichiers (y compris les fichiers cachés) et des détails sur les fichiers, on peut par exemple utiliser la commande « `ls -al` » (voir « `man ls` » pour plus d'informations).

5. La commande **touch** permet de créer un fichier vide ou, si le fichier existe, de le « toucher », c'est à dire d'y accéder, ce qui change la date du dernier accès au fichier. Exemple, « `touch exo` » crée un fichier exo si celui-ci n'existe pas.

6. La commande **cp** (CoPy) permet de copier des fichiers ou des répertoires. Exemple :

   « `cp exo ~/Documents` » copie le fichier `exo` dans le répertoire `Documents` du répertoire personnel.

7. La commande **mkdir** (MaKe Directory) permet de créer un répertoire.
   Exemple : « `mkdir NouveauRep` ».
8. La commande **mv** (MoVe) permet de déplacer ou renommer un fichier. Exemple :
   « `mv exo NouveauRep/monexo` » déplace le fichier `exo` dans le répertoire `NouveauRep` en le renommant `monexo`.
9. La commande **find** permet de rechercher des fichiers. Exemple : « `find -name monexo` ».
10. La commande **rm** (ReMove) est une commande risquée puisqu'elle permet de supprimer un fichier. Celui-ci est totalement supprimé, il ne passe pas par la « corbeille », et selon la configuration de la console, elle ne demande pas de confirmation. Exemple : « `rm -R NouveauRep` » supprime le répertoire `NouveauRep` ainsi que (récursivement) son contenu.

## 2. Gestion des droits et permissions

On peut voir, en utilisant la commande « `ls -l` » que pour chaque fichier il y a des informations de la forme « `-rw-r--r--` » ou bien « `drwxr-xr-x` ». Cela indique les droits associés au fichier (si cela commence par '-') ou ou répertoire (ci cela commence par 'd'), sous forme de trois fois trois caractères. Dans chacun des trois triplets de caractères, on a dans l'ordre, 'r' (read) pour la lecture, 'w' (write) pour l'écriture et 'x' pour l'accès/ouverture (pour un répertoire) ou l'exécution (pour un fichier).

- Le premier triplet correspond aux droits de l'utilisateur possédant le fichier
- Le second triplet correspond aux droits des membres du groupe auquel appartient le fichier
- Le troisième triplet correspond aux droites des autres utilisateurs

Exemple : un fichier ayant les droits « `-rw-r--r--` » donne les droits de lecture à tout le monde, mais les droits d'écritures ne sont donnés qu'à l'utilisateur qui possède le fichier. Le fichier n'est cependant pas exécutable.
Précisons : chaque fichier (et répertoire) est la propriété d'un utilisateur particulier. Par défaut, celui-ci appartient à l'utilisateur qui a créé le fichier. Les utilisateurs sont réunis en groupe. Un utilisateur pouvant faire partie de plusieurs groupes, pour chaque fichier est spécifié le groupe propriétaire (c'est-à-dire en tant que membre de quel groupe le propriétaire détient le fichier). On distingue alors trois catégories d'utilisateurs pour chaque fichier : le propriétaire, le groupe propriétaire, et les autes. Comme nous l'avons indiqué plus haut, les droits du fichier sont donc définis pour chacune de ces catégories.
Avant de manipuler ces droits, indiquons une commande supplémentaire, permettant de connaître notre identité. Il s'agit de la commande « **id** ». Elle donne les numéros et noms d'utilisateur (uid), de groupe (gid) et la liste des groupes auxquels on appartient.

**Exercice 2**

1. Déterminer vos groupes d'appartenance à l'aide de la commande **id**.
2. Aller dans le répertoire personnel, utiliser la commande « `ls -l` » et repérer les droits, le propriétaire et le groupe d'appartenance de chacun des fichiers qui s'y trouvent.

Pour changer les droits (si on en a les droits suffisants), on utilise la commande **chmod**.
La syntaxe générale est « `chmod liste_droits fichier(s)` ». La liste des droits peut être par exemple :

- `u+r` pour rajouter au propriétaire (user) le droit en lecture,
- `g-w` pour retirer aux membres du groupe (group) le droit en écriture,
- `o+x` pour donner aux autres utilisateurs (other) le droit en exécution,
- ou une combinaison de ces possibilités (exemple : `ug-wx`).

À l'aide de la commande « `echo "une phrase" > fic` », on peut écrire le texte « une phrase » dans le fichier `fic`.

**Exercice 3**

1. Créer un répertoire test, et un fichier essai dans ce répertoire, et y écrire la phrase de votre choix.
2. Notez à l'aide de `ls -l` les permissions actuelles du répertoire test et du fichier essai.
3. En utilisant la commande `chmod`, retirez-vous le droit en lecture et en écriture sur le fichier essai. Vérifier l'effet obtenu en essayant d'afficher le contenu du fichier sur la fenêtre du terminal (avec la commande « `cat essai` » par exemple), puis de remplacer ce contenu par une phrase différente.
4. Un fichier exécutable est simplement un fichier dont on possède le droit en exécution. Rétablir le droit en écriture puis remplacer à l'aide de la commande `echo` le contenu du fichier essai par le texte « `echo "Ceci est un essai"` ». Ajoutez-vous le droit en exécution, et exécuter le fichier essai en tapant « `./essai` dans le terminal (depuis le répertoire qui le contient). Quel est le problème ?
5. Rétablir enfin le droit en lecture et tenter à nouveau d'exécuter le fichier.

   Si tout ne se passe pas vraiment comme prévu, comment obtenir un résultat plus intéressant ?
6. Modifier enfin les droits du fichier `essai` afin qu'un membre du groupe propriétaire puisse le modifier, puis que les autres ne puissent pas le lire.

Passons maintenant à la manipulation des droits sur les répertoires :

**Exercice 4**

1. Se placer dans le répertoire `test`, et retirez-vous le droit en lecture pour ce répertoire (à savoir : le répertoire courant est le répertoire « . », ce qui est pratique si on ne veut pas donner son adresse absolue). Lister le contenu du répertoire avec `ls`, puis exécuter ou afficher le contenu du fichier `essai`. Que peut-on en déduire ? Rétablir le droit en lecture sur `test`.
2. Créer dans `test` un fichier `nouveau` ainsi qu'un répertoire `sstest`. Retirer au fichier `nouveau` et au répertoire `test` le droit en écriture. Tenter de modifier le fichier `nouveau`, puis de le supprimer. Rétablir ensuite le droit en écriture au répertoire `test`. Tenter de modifier le fichier `nouveau`, puis de le supprimer. Que peut-on déduire de toutes ces manipulations ?
3. Se positionner dans le répertoire personnel, puis retirer le droit en exécution du répertoire `test`. Tenter de créer, supprimer, ou modifier un fichier dans le répertoire `test`, de s'y déplacer, d'en lister le contenu, etc... Qu'en déduit-on quant au sens du droit en exécution pour les répertoires ?