Commandes Linux

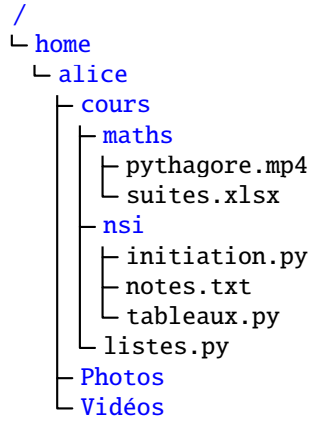
Le terminal Linux

Dans Linux, comme dans Unix, tout est un fichier. C'est un de ses principes de base. Ainsi, les fichiers sont des fichiers, les dossiers sont des fichiers, les processus sont des fichiers et même les périphériques sont des fichiers. C'est-à-dire qu'ils sont tous accessibles dans une arborescence dans laquelle on peut se déplacer au sein d'un terminal, à l'aide de lignes de commandes.
Ces commandes sont interprétées par un programme appelé **shell** et transmises au système d'exploitation. Le shell est le nom générique de ces interpréteurs. Dans le cas de Linux, il y a plusieurs interpréteurs, mais le plus courant est `bash`.

Déplacement dans l'arborescence

L'arborescence des fichiers est organisée comme un arbre dont la racine se note '`/`'. Dans chaque dossier on trouve un dossier '`.`' qui correspond au dossier actuel et '`..`' qui correspond au dossier parent du dossier actuel.
Pour vous déplacer dans l'arborescence des fichiers, il faut utiliser la commande '`cd DEST`' pour aller dans '`DEST`'. La destination '`DEST`' peut être :

- une **adresse absolue** : On indique le chemin depuis la racine, comme `/home/alice/Vidéos`.
- une **adresse relative** : On indique le chemin à prendre pour aller du dossier actuel à la destination. Pour remonter dans l'arborescence il faut utiliser '`..`'.
  Par exemple, on peut faire '`cd ../../Photos`'.
- rien du tout : S'il n'y a pas de destination, on va directement au dossier principal de l'utilisateur. On peut aussi aller dans ce dossier en faisant '`cd ~`'.



Exercice 1 : On considère l'arborescence ci-contre.
1) Donner l'adresse absolue du dossier `nsi`.
2) Donner l'adresse relative pour aller de `Photos` à `nsi`.
3) De quel dossier peut-on venir si on fait '`cd ../../Photos`'?

Pour se repérer dans les dossiers, vous pouvez utiliser les commandes suivantes :
- '`pwd`' affiche le chemin absolu permettant d'arriver au dossier courant.
- '`ls`' affiche tous les fichiers et dossiers se trouvant dans le dossier courant. On peut également rajouter une ou des adresses relatives ou absolues pour lister leur contenu.

Exercice 2 : La commande '`pwd`' a renvoyé '`/home/alice/cours`'. Quelle liste de dossiers et fichiers va être obtenue avec la commande :
1) `ls` 2) `ls maths` 3) `ls ..`

La commande '`ls`' possède de nombreuses options, comme la plupart des commandes Linux. Ainsi, '`ls -a`' permet d'afficher les dossiers et fichiers cachés, c'est-à-dire ceux qui commencent par un point, comme '`.bashrc`' ou '`.config`'. L'option '`-l`' permet d'obtenir de nombreuses informations sur les fichiers et dossiers comme les droits et la date de dernière modification. Dans la plupart des distributions Linux, on peut utiliser l'**alias** '`ll`' à la place de '`ls -l`'.

Création ou suppression de dossiers

Si on veut rajouter un nouveau dossier dans le dossier courant, on peut faire '`mkdir DOSS`'. On peut également créer plusieurs dossiers en même temps. Ainsi '`mkdir d1 d1/d2 d1/d2/d3`' va créer un dossier '`d1`' dans lequel sera créé '`d2`' dans lequel sera créé '`d3`'. Pour créer un sous-dossier, il faut que le dossier parent existe déjà.
L'option '`-p`' permet de créer tous les dossiers nécessaires. Ainsi, '`mkdir -p d1/d2/d3`' va créer les dossiers '`d1`', '`d2`' et '`d3`' s'ils n'existent pas déjà.
Pour supprimer un dossier, il faut qu'il soit vide et ensuite faire '`rmdir DOSS`'. Là encore, on peut en supprimer plusieurs à la suite.

Manipulation des fichiers

Une fois qu'on sait se déplacer dans les dossiers, il faut apprendre à manipuler les fichiers. Pour créer un fichier vide, il suffit de faire : '`touch fichier`', ce qui va créer un fichier vide appelé '`fichier`' dans le dossier courant. Il est également possible d'en créer plusieurs à la suite. Ainsi '`touch doss/f1.txt ../f2.txt f3.txt`' va créer '`f1.txt`' dans le dossier '`doss`', '`f2.txt`' dans le dossier parent et '`f3.txt`' dans le dossier courant. De façon générale, toutes les commandes suivantes permettent de manipuler des fichiers se trouvant dans d'autres dossiers en indiquant le chemin relatif ou absolu pour y arriver.
Pour supprimer un fichier il faut faire '`rm FICH`'. Il est possible d'en mettre plusieurs à la suite. La commande '`rm -r DOSS`' permet également de supprimer un dossier '`DOSS`' et son contenu s'il n'est pas vide.
Pour déplacer ou renommer un fichier, il faut faire '`mv FICH DEST`'. Si '`DEST`' est un dossier existant, alors le fichier sera déplacé dans ce dossier. Par contre, si c'est un nom de fichier, alors le fichier '`FICH`' va être renommé en '`DEST`' et déplacé si ce n'est pas dans le même dossier. Cette commande permet aussi de déplacer ou renommer un dossier.
De la même manière la commande '`cp FICH DEST`' permet de copier un fichier. Si '`DEST`' est un dossier existant, un fichier avec le même nom que '`FICH`' sera copié dans '`DEST`'. Sinon le fichier copié s'appellera '`DEST`'. Pour copier un dossier, il faut rajouter l'option '`-r`'.
Par exemple, la commande '`cp d1/fich1.txt d2/fich2.txt`' va copier le fichier '`fich1.txt`' qui se trouve dans le dossier '`d1`' dans le dossier '`d2`' et l'appeler '`fich2.txt`'.

Lire et modifier un fichier

La commande '`cat`' est le moyen le plus simple pour afficher le contenu d'un fichier. Ainsi '`cat FICH`' affiche le contenu de '`FICH`' dans le terminal. Si on donne le nom de plus d'un fichier, ils sont tous affichés les un après les autres.
Il y a de nombreuses commandes permettant de voir le contenu d'un fichier. Par exemple '`head FICH`' et '`tail FICH`' affichent respectivement les premières et dernières lignes du fichier. Pour pouvoir se déplacer dans le fichier ou faire des recherches, il y a la commande '`less FICH`'. Vous pouvez vous déplacer avec les touches ↑ et ↓. Vous pouvez chercher un mot ou du texte en tapant '`/TEXTE`'. S'il y a plusieurs résultats dans le fichier d'aide, la touche '`n`' vous permettra de passer à la prochaine occurrence ou '`shift+n`' pour revenir à la précédente. Pour quitter, il faut faire '`q`'.
Enfin, il y a de très nombreux éditeurs de textes en ligne de commande permettant de modifier le fichier sans interface graphique. On peut citer, entre autres, '`nano`', '`ed`', '`vi`', '`vim`' ou encore '`emacs`'.

Exercice 3 : On suppose que l'on commence dans un dossier 'exercice' vide.

```
mkdir -p doss1/doss2/doss3
cd doss1
mkdir -p ../doss2/doss3
touch f1.txt doss2/doss3/f2.txt
```

```
cp -r doss2/doss3 ../doss4
cd ../doss4
mv ../doss1/f1.txt .
cp f1.txt ../doss2/f3.txt
```

Quel est le contenu du dossier 'exercice' après ces commandes ?

Gérer les droits

Les systèmes Linux, en tant qu'héritiers d'Unix, sont multi-utilisateurs. Il faut donc pouvoir déterminer qui peut lire, modifier ou exécuter les différents fichiers ou dossiers. Pour cela on distingue 3 types d'utilisateurs : le propriétaire du document, les utilisateurs du même groupe et les autres. Chaque utilisateur est membre d'un ou plusieurs groupes. En général, lors de la création d'un utilisateur, un nouveau groupe, du même nom est aussi créé. Mais d'autres groupes sont plus génériques. On pourrait créer un groupe 'prof' qui permettrait à tous les professeurs de travailler ensemble sans que les élèves ne puissent y accéder.
La commande 'ls -l' permet de connaître qui a le droit de faire quoi avec les différents fichiers ou dossiers dans le dossier courant. Voici un extrait de réponse :

```
-rw------- 1 alice alice   93  5 mars  15:10 aa.txt
-rw-r--r-- 1 alice alice   96  5 mars  15:10 ab.txt
-r--r--r-- 1 alice alice   76  5 mars  15:10 bc.txt
-rw--w---- 1 alice alice  119  5 mars  15:10 bd.txt
drwx--x--x 3 alice alice 4096  5 mars  15:10 dossier1
-rwxr-xr-x 1 alice alice   74  5 mars  15:10 script.sh
```

Pour chaque objet, cela affiche les droits, le nombre de liens pointant sur cette ressource, le propriétaire, le groupe propriétaire, la taille (en ko), la date et l'heure de la dernière modification et le nom.
Les droits sont composés de 10 symboles. Le premier est spécial et sert juste à distinguer les dossiers des fichiers. Il y a ensuite 3 blocs de trois symboles 'rwx' pour *lire* (read), *écrire* (write) et *exécuter* (execute). Un '-' signifie que le droit n'est pas attribué. Les 3 triplets correspondent aux droits du propriétaire, du groupe et des autres. Ainsi, le fichier 'bd.txt' peut être lu et modifié par alice, peut être seulement modifié par les autres membres du groupe alice et les autres utilisateurs ne peuvent rien faire avec ce fichier. Personne ne peut l'exécuter, contrairement à 'script.sh' que tout le monde peut exécuter.
La commande chmod permet de modifier les droits d'un fichier ou dossier. Seul le propriétaire, ou un administrateur, peut changer les droits d'un document. Cette commande s'utilise de deux façons différentes. On peut explicitement indiquer les modifications à effectuer. On indique d'abord qui est concerné : 'u' (*user* : utilisateur), 'g' (*group* : groupe), 'o' (*other* : autre utilisateurs) ou 'a' (*all* : tous). Ensuite, on indique si on veut rajouter un droit (+) ou l'enlever (-). Puis on met le ou les droits concernés. On peut séparer plusieurs actions par des virgules. Voici quelques exemples :

- `chmod u+x,o-r FICH` : donne le droit d'exécuter à l'utilisateur et retire le droit d'écriture aux autres (hors groupe).
- `chmod ug+rw FICH` : donne le droit d'écrire et de modifier à l'utilisateur et à tous les membres du groupe.
- `chmod o+r-w FICH` : donne le droit lire mais pas d'écrire aux autres (hors groupe).
- `chmod +x FICH` : donne le droit d'exécuter à tout le monde. Sans précision de qui est concerné, c'est 'a' qui est utilisé.

Exercice 4 : En partant des fichiers listés plus haut, déterminer les droits attribués aux fichiers après les commandes suivantes :

1) `chmod g+rw,o+w aa.txt`
2) `chmod g+w,o-x script.sh`

Les droits sont aussi identifiés à un nombre de 3 chiffres correspondant à 3 nombres binaires de 3 bits écrit en octal (base 8). Chaque bit correspond à un droit : 1 si le droit est atribué et 0 sinon.

	user			group			other		
droits	r	w	x	r	w	x	r	w	x
valeur	4	2	1	4	2	1	4	2	1
exemple	r	w	-	r	-	-	-	-	-
octal	6			4			0		

La commande `chmod 640 FICH` donne donc les droits `rw- r-- ---` à `FICH`.

Exercice 5 :

1) Déterminer le code octal des droits suivants :
   a) `rw- -w- r--` b) `rwx r-x --x` c) `r-x r-- ---`
2) Déterminer les droits correspondant aux codes suivants :
   a) 654 b) 732 c) 410

Aide sur les commandes

Les commandes Linux possèdent de très nombreuses options. Nous en avons déjà vu plusieurs, comme '`ls -l`', '`mkdir -p`' ou '`rm -r`'. Pour obtenir la liste des options, de leurs effets et plus généralement des explications sur le fonctionnement de la commande, il y a généralement deux possibilités. La plupart des commandes ont une option '`-h`' ou '`--help`' qui permet d'afficher une aide succinte. Pour avoir le manuel d'une commande, il faut faire `man COMMANDE` qui ouvre la documentation. Les contrôles sont les mêmes que ceux de '`less`'.
Une fois que vous avez trouvé les options que vous voulez utiliser, par exemple '`-l`' et '`-a`' pour '`ls`', vous pouvez les mettre séparément ou les coller : '`ls -l -a`' ou '`ls -la`'.

Le GLOB

Les motifs GLOB permettent de manipuler en même temps plusieurs fichiers ou dossiers dont les noms correspondent à certaines contraintes. Ces motifs se basent sur deux caractères spéciaux : '`?`' et '`*`'. Le premier permet de remplacer n'importe quel symbole (ou presque) et le deuxième permet de remplacer une suite de 0, 1 ou plusieurs caractères. Chaque motif est ensuite remplacé par la liste de tous les fichiers et dossiers qui correspondent. Par exemple '`co*`' peut être remplacé par '`co`', '`cours`', '`copie.txt`' ou '`comptes.odt`', mais pas '`ocultation.py`' ou '`deco.jpg`'.
Il y a deux limitations : '`?`' et '`*`' ne peuvent pas remplacer un '`/`' ou un point en début de nom. Cela signifie que '`mv *.py python/`' permet de déplacer tous les fichiers Python du dossier courant dans le dossier '`python`' mais pas ceux d'autres dossiers, ni '`.cache.py`'.
Des extensions des motifs permettent de définir des plages de symboles ou d'en exclure.

Exercice 6 : Pour chacun des motifs suivants, donner deux exemples de noms de dossiers ou de fichiers qui peuvent correspondre :

1) `a?a.py` 2) `*/*.jpg` 3) `202?-202?` 4) `.*.sh` 5) `a?b*`

Pour aller plus loin

Les commandes Linux permettent de faire bien plus que ce qui a été présenté ici. Il est possible de définir des variables, de faire des boucles ou des tests. En fait, on peut écrire des programmes, appelés **scripts**, à l'aide de ces commandes. Ces scripts sont particulièrement utiles pour administrer un système Linux et pour automatiser certaines tâches.
Mais les commandes Linux permettent de faire encore plus. Les **redirections** servent à rediriger le résultat d'une commande dans un fichier. Par exemple '`ls > listing.txt`' renvoie le résultat de '`ls`' dans le fichier '`listing.txt`'. Les **pipes** permettent eux de rediriger le résultat d'une commande comme paramètre d'une autre commande. Ainsi, '`ls | head`' n'affiche que les premiers fichiers et dossiers. Couplées entre elles et avec des commandes comme `grep`, qui permet de chercher dans le contenu des fichiers, ou `sed` qui permet de modifier ce contenu, les possibilités sont immenses.