Un brève introduction au terminal linux

Mon ami le terminal

Pour naviguer dans l'arborescence des dossiers et agir les fichiers de façon plus efficace on utilise un terminal, dans lequel on tappe des commandes au lieu de l'interface graphique habituelle. Voici quelques notions et commandes utiles à l'utilisation d'un terminal.

Notion de chemin

Un chemin définit l'emplacement d'un dossier dans l'arborescence du disque dur. Le `/` définit l'origine de l'arborescence, c'est à dire, la racine du disque dur (`C:\` sous windows). On donne ensuite la liste des dossiers successifs, séparé par des `/`, jusqu'à celui que l'on désire. Dans l'exemple suivant on donne l'emplacement du fichier `proteine.pdb` qui est dans le dossier `CFP` lui-même dans le dossier `user` qui est dans le dossier `home` qui est à la racine.

```
/home/user/CFP/proteine.pdb
```

Quand on donne l'emplacement d'un fichier par rapport à la racine, le chemin commence par `/`. On dit que c'est un chemin *absolue*. On peut utiliser un chemin *relatif* dans lequel on donne la succession des dossiers à partir du dossier dans lequel on se trouve. Un chemin relatif dépend donc du dossier dans lequel on se trouve.

Si on est dans le dossier `/home/user/CFP/` le fichier `proteine.pdb` est dans le dossier dans lequel on se trouve. Son chemin relatif est donc tout simplement `./proteine.pdb`. Le `.` signifie "ici", il désigne l'endroit où on se trouve. Le chemin relatif du dossier `user` qui est le dossier parent du dossier `CFP` est `../user/`. Les deux points `../` signifient "parent", ils désignent le dossier qui contient celui dans lequel on se trouve.

l'étoile *

En ligne de commande, on peut agir simultanément sur plusieurs fichiers grace au caractère *. L'étoile remplace tous les caractères. Par exemple `cp calcul* ../` copiera tout les fichiers du dossier courant qui commence par `calcul` dans le dossier parent quelque soit la fin du nom des fichiers. Si on fait `cp * ../` on copie l'ensemble des fichiers du dossier courant dans le dossier parents.

Attention, l'usage des étoiles peut être dangereux notamment dans le cas de l'utilisation de la commande `rm` si on fait une erreur de frappe. Par exemple si l'on tappe `rm calcul *`, à cause de l'espace, on va supprimer le fichier `calcul` et le fichier `*` c'est à dire **TOUS** les fichiers contenus dans le dossier courant !!!! Tout ça pour un espace en trop !

Un petit mot sur les permissions

Sous linux chaque personne a un nom d'utilisateur `user` et appartient a un ou plusieurs `groupe`. Si on tappe la commande `ll`, le nom des fichiers est précédé d'un tas d'informations :

```
drwxr-xr-x 2 user cth 4,0K oct 27 15:48 Desktop
-rw-r--r-- 1 user cth  31K oct 31 11:02 toto
```

De droite à gauche, il y a le nom du fichier la date de dernière modification et la taille du fichier. Puis on trouve `cth` qui est le groupe de l'utilisateur qui est propriétère du fichier et `user` qui est le nom de cet utilisateur. Le dernier élément donne les permissions associées au fichier ou au dossier. On a d pour directory, r pour read, w pour write, et x pour exécutable. Le premier tirer nous dit s'il sagit d'un fichier ou d'un dossier. Ensuite les tirets vont par trois. Les trois premiers donnent les permissions de l'utilisateur (r w et/ou x), puis celles du groupe (r w et/ou x) et enfin celles des autres personnes.

Dans cet exemple, `toto` est un fichier appartenant à user du groupe cth. L'utilisateur peut le lire et écrire dedans et les autres membres du groupe cth ou des autres groupe peuvent lire le contenu du fichier mais pas le modifier.

Lorsqu'on compile un programme, il crée un fichier binaire que l'on doit exécuter pour éxécuter le programme. Pour cela il faut avoir la permission d'exécuter ce fichier (il faut qu'il y ait un x). Si ce n'est pas le cas on utilise la commande `chmod +x fichier` pour rendre le fichier exécutable.

Les noms des fichiers et la casse des caractères

Il est conseillé de n'utiliser que les 36 caractères alphanumériques et le `.` le `-` ou le `_` pour nommer des fichiers ou des dossiers. Entre autre, il est conseillé de ne pas utiliser d'espaces, de caractères accentués ou d'autres symboles. Cette limitation n'est pas due à linux mais à l'utilisation du terminal.

La casse des caractères est la différenciation entre les minuscules et majuscules. Linux fait la différence entre les majuscules et les minuscules alors que Windows n'en fait pas. Par exemple les fichiers `ToTo` et `toto` sont identique sous windows mais différents sous linux. Cette différence est également très importante pour les mots de passe.

Quelques Commandes de bases

Syntaxe	Résultat
`ls`	Affiche le contenu du dossier courant.
`ls -l`	Affiche le contenu du dossier courant sous forme de liste.
`cd chemin`	**C**hange **D**irectory : on se déplace en suivant le chemin donné.
`.`	représente le dossier dans lequel on se trouve
`..`	représente le dossier au dessus (dossier parent)
`cd ../`	on va dans le dossier parent. Si on reprend l'exemple, si on est dans le dossier `CFP`, on va dans le dossier `user` en faisant `cd ../` et dans le dossier `home` en faisant `cd ../../`
`cp nom1 nom2`	**C**o**P**y. Permet de copier un fichier. Par exemple `cp fichier1 fichier2` créera `fichier2` identique au `fichier1`. Autre exemple `cp /usr/bin/programme/exe .` copiera le fichier `exe` qui se trouve dans `/usr/bin/programme` dans le dossier courant (le point ".").
`mv nom1 nom2`	**M**o**V**e. Permet de renommer ou déplacer un fichier ou un dossier. Exemple `mv tata toto` remplace le fichier tata par toto. Autre exemple : `mv toto ../` déplace le fichier toto dans le dossier parent.
`rm nom`	**R**e**M**ove. Supprime le fichier nom. ATTENTION : 1) il n'y a pas de corbeille dans le terminal 2) être prudent avec les étoiles.
`mkdir nom`	**M**a**K**e **DIR**ectory. Crée un dossier nom dans le dossier courant.
`vi nom`	Ouvre le fichier nom avec l'éditeur vim pour le lire ou le modifier. Vim est un éditeur de fihcier texte.
`gvim nom`	Ouvre le fichier nom avec l'éditeur gvim pour le lire ou le modifier. gvim est l'équivalent de vim mais dans une interface graphique.
`pwd`	**P**rint **W**ork **D**irectory, affiche le chemin absolue du dossier courant.
`xm` ou `xmgrace`	xmgrace est un programme qui permet de tracer des courbes, faire des graphiques ect ... Si on tappe `xm monfichier`, le programme trace en X la première colonne et en Y la deuxième. `xm -nxy fichier` trace en X la première colonne et en Y autant de courbes qu'il y a de colonnes.
`grep`	Permet de faire une recherche dans un fichier. Par exemple `grep "SCF Done" calcul.log` affichera la ou les lignes du fichier `calcul.log` qui contiennent l'expression `SCF Done`.
`gfortran`	Gfortran est un compilateur de fortran77/90. Il crée un fichier `a.out` qui est le fichier exécutable. Pour exécuter le programme on tappe `a.out` dans le terminal.
`ifort`	Ifort est le compilateur de fortran77/90 de intel. Il crée un fichier `a.out` qui est le fichier exécutable. Pour exécuter le programme on tappe `a.out` dans le terminal.
`diff`	**DIFF**érence. `diff fichier1 fichier2` affichent les lignes qui sont différentes dans les deux fichiers. Des variantes sont `vimdiff` ou `gvimdiff` qui affiche les deux fichiers côte à côte en surlignant les parties qui sont différentes dans les deux fichiers.
`man commande`	Affiche le manuel de la commande que l'on a donnée.
`history`	Affiche l'historique des commandes tappées dans le terminal.
`qsub job`	soumettre le calcul `job` à la grappe de calcul.
`qstat`	Affiche la liste des calculs en cours.
`qstat -q`	Affiche la liste des queues, leurs caractéristiques, et le nombre de calculs en cours.
`qstat -u user`	Affiche la liste des calculs soumis par l'utilisateur `user`.
`qsub -W depend=afterok:XXXXX.styx job`	soumet le calcul `job` dans un état Held, c'est à dire en attente. Il démarera lorsque le calcul qui porte le numéro XXXXX.styx sera fini.

A voir également, la page du site web du labo concernant l'utilisation de la grappe : `http://www.lcp.u-psud.fr/rubrique.php3?id_rubrique=193`

Dans un terminal, si on lance une application le terminal attend qu'elle soit finie avant que l'on puisse en lancer une autre ou tapper une commande. Pour tapper d'autres commandes dans le même terminal sans fermer l'application il faut, lorsqu'on la lance, la faire suivre d'un `&`, par exemple `xmgrace &`.

L'éditeur Vim

Vim est un éditeur de texte pratique car il est rapide et il possède de nombreuses fonctions permettant d'agir rapidement sur des groupes de lignes ou de mots. Il se démarque des

éditeur de texte classique par le fait que l'écriture (l'édition) n'est pas active par défaut. Ainsi, le clavier peut servir soit à tapper des commandes pour agir sur le texte, soit pour écrire du texte. On peut utiliser Vim en mode Insertion/remplacement qui est le mode utilsié pour écrire le texte, ou en mode normal dans lequel le clavier est utilisé pour tapper des commandes. Pour écrire dans le fichier il faut donc commencer par se mettre en mode Insertion avec la touche i (ou la touche inser du clavier).

En bas du fichier sont affichées des indications comme le mode dans lequel on est (insertion/remplacement ou normal) ainsi que la ligne et la colonne sur laquelle se trouve le curseur. La touche Echap doit être utilisée sans modération pour revenir en mode normal (ni insertion ni remplacement). Avant de tapper toute commande il vaut mieux faire Echap (au cas ou) !!

Quelques Commandes de bases

Syntaxe	**Résultat**
`i`	Passe en mode **i**nsertion, pour commencer à écrire. On peut aussi utiliser la touche "Inser" du clavier.
`r`	Passe en mode **r**emplacer. On écrit en écrasant ce qui se trouve sous le curseur. Par défaut on ne remplace qu'une seule lettre. Pour bloquer le mode remplacement il faut faire **R**. La touche "Inser" fait basculer du mode insertion au mode remplacement.
`u`	**u**ndo. Lorsqu'on est ni en mode insertion ni remplacement, annule la dernière modification. Si on annule une modification on ne peut pas la rétablir.
`Echap`	Sort du mode insertion ou remplacement
`x`	Lorsqu'on est ni en mode insertion ni remplacement, efface le caractère sous le curseur.
`dd`	Lorsqu'on est ni en mode insertion ni remplacement, efface la ligne actuelle sur laquelle se trouve le curseur.
`:w` (nom)	Sauvegarder ou sauvegarder dans le fichier "nom" s'il est donné.
`:q`	quitter sans sauvegarder.
`:q!`	quitter sans sauvegarder et en ignorant les modifications effectuées depuis la dernière sauvegarde.
`:wq`	Enregistrer et quitter.
`:r` "nom"	inclu le fichier nom à la suite du curseur.
`Y`	**Y**ank copie la ligne sur laquelle est le curseur.
`p` ou `P`	**P**aste. Colle la ligne précédement copier avec Y sur la ligne au dessus ou en dessous.
`o` ou `O`	Crée une nouvelle ligne au dessus ou en dessous de la ligne courante et passe en mode insertion.
`A`	Passe en insertion à la fin de la ligne courante.
`D`	Efface la fin de la ligne courante.
`cw`	Efface le mot sur lequel est le curseur et passe en mode insertion.
`/ expression`	Recherchera le mot ou l'expression dans le fichier et les surlignera si vim les trouves.
`:1`	Aller à la première ligne du fichier.
`:$`	Aller à la dernière ligne du fichier.

Utilisation du .

Sous vim lorsqu'on écrit ou efface un mot ou une expression il est possible de le répéter autant de fois que l'on veut avec la touche `.` du clavier. Par exemple si l'on supprime les 10 premiers caractères d'une ligne, on sort du mode insertion avec echap, puis à chaque utilisation de la touche `.` sur une nouvelle ligne ou sur la ligne courante, supprimera le même nombre de caractères.

Agir sur plusieurs lignes

Pour agir sur plusieurs ligne, on utilise leur numéro. Par exemple pour effacer les lignes de 10 à 20 on tappe la commande

```
:10,20 d
```

On peut utiliser le dollar pour spécifier la dernière ligne du document ou le . pour spécifier la ligne sur laquelle se trouve le curseur. On peut "marquer" une ligne, pour sauvegarder son numéro dans une variable avec la lettre `m`. Par exemple si on tappe `m a` sur une ligne et `m b` sur une autre, on peut copier les lignes entre celles que l'on vient de marquer par

```
:'a,'b Y
```

où `'a` et `'b` représentent les numéros des lignes que l'on a marquées.

Pour simplement supprimer 5 ligne à partir du curseur on tappe simplement `5 dd` ou pour les copier `5 Y`. De manière générale si on tappe un chiffre suivi d'une commande, elle est répété autant de fois que le chiffre tappé avant.