Linux, ce qu'il faut savoir pour les examens

Les systèmes d'exploitations basés sur Linux ont a peu près tous les mêmes commandes, et la même arborescence de fichier. Les systèmes linux les plus connus sont sans doutes Ubuntu, Android et Debian d'où est repris le système d'exploitation du Raspberry Pi, Raspbian.

Système de fichier

Le système de fichier de Linux à une arborescence qui ne part pas du disque (comme Windows), mais de la base du système de fichier appelée `root`. Ensuite, vous avez un certain nombre de dossiers prédéfinis, qui sont les suivants (liste exacte dépendante du système d'eploitation):

- `bin`
- `boot`
- `etc`
- `usr`
- `var`
- `home`
- `lib`
- `mnt`
- `media`
- `opt`

- `proc`
- `root`

`/bin`

Le dossier `/bin` contient une partie des programmes et applications livrés avec le système. C'est aussi là que se trouvente les défintions de quelques commandes de base de Linux, comme par exemple `cd`.

`/boot`

Le dossier `/boot` contient tous les fichiers nécessaires au démarrage du système d'exploitation. Il est fortement recommandé de ne pas le toucher du tout.

`/etc`

Le dossier `/etc` est votre centrale de configuration. En effet, c'est là que se trouvent les fichier de confguration de votre système et de certaines des applications installées. Vous pouvez auusi trouver dans ce dossier certains de vos mots de passe, ou alors des informations sur votre réseau (imprimantes ou ordinateurs connectés en réseau).

`/home`

Le dossier `/home` contient les différents dossiers des utilisateurs, qui contiennent eux-mêmes vos dossiers d'image, de téléchargements, de documents ou votre bureau. Il est équivalent au dossier `users` ou `utlisateurs` de Windows.

`/mnt` et `/media`

Les dossiers `/mnt` et `/media` servent tout les deux à monter tous les disques amovibles connectés à l'ordinateur, que ce soit des clés USB ou des disques durs externes. De cette manière l'on accède pas à un disque externe en cherchant un nouveau disque dans la lisate des disques disponibles (comme sur Windows) mais en se rendant dans un de ces dossiers, et en prenant le nom du disque amovible comme nom de dossier. La grosse différence entre ces deux dossiers, est que le dossier `/media` ne recoit que les disques montés de manière automatiques (la plupart des cas), tandis que le dossier `/mount` ne contiendra que les disques montés manuellement ou les disques auxquelles l'utilisateur a attribué des paramètres spéciaux.

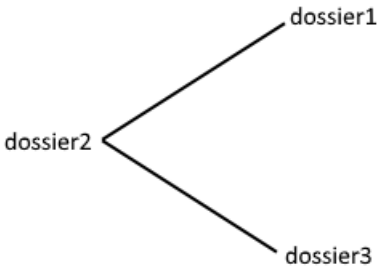
Les commandes

Avant l'utilisation des interfaces graphiques (GUI), l'utilisation des commandes étaient obligatoire. Avec la démocratisation de l'informatique, les interfaces graphiques sont devenues la norme, car ils sont plus clairs et donc plus accessibles au débutants. Cependant, les commandes de sont pas rester sur le carreau, puisqu'elles permettent de gagner de l'efficacité dans certaines actions toute en consommant nettement moins de ressources que les interfaces graphiques. Les commandes sont à utiliser de la manière suivante, `commande <paramètres> <arguments>`. Les commandes étudiées sont les suivantes:

- `cd`
- `chmod`
- `chown`
- `cp`
- `ls`
- `mkdir`
- `mv`
- `nano`
- `pwd`
- `rm`
- `su` et `sudo`
- `shutdown`

`cd`

La commande `cd` sert à changer de répertoire (dossier). Pour arriver dans une dossier en entrant son chemin absolu, il suffit de rentrer ce chemin en faisant attention de bien commencer par un slash. Par exemple, un dossier1 et un dossier3 contenus dans le dossier2 qui se trouve à la racine serait accédé par la commande absolue `cd /dossier2/dossier1`.



Mais certaines fois il est plus utile d'utiliser les chemins relatifs, par exemple si vous êtes dans un dossier passablement éloigné de la racine et que vous voulez juste aller dans un sous-dossier. Un chemin relatif est un chemin qui va partir du dossier actuel. Ainsi si nos 3 dosssiers précédents étaient très éloigné de la racine et que nous nous trouvions dans le dossier2, il nous suffirait juste de taper `cd dossier1` pour accéder au dossier1. À noter que je n'ai pas utilisé de slash, puisque le chemin est relatif. Maintenant vous savez comment accéder à un dossier en absolu, comment monter dans l'arborescence avec les chemins relatifs (s'éloigner de la racine), il est temps de voire comment se rapprocher de la racine. C'est tout simple , il suffit juste d'utiliser `..` au lieu du nom de dossier. Ainsi pour passer du dossier 1 au dossier 2 en relatif, il suffit d'utiliser `cd ..`. Et ce n'est pas tout, car il est aussi possible de les combiner avec d'autres chemin relatifs. Ainsi si on voulait passer de dossier1 au répértoire parent de dossier2, il suffit d'utiliser `cd ../..`. Ou alors, si on veut passer de dossier1 à dossier3, on tape `cd ../dossier3`. Pour finir, le sigle environ (~) permet d'utiliser le répertoire de l'utilisateur comme répertoire de base.

`cp`

La commande `cp` sert à copier des fichiers ou des dossiers. La commmande demande un fichier ou dossier source qui sera le premier entré, et un dossier ou un fichier de destination qui sera le second chemin entré. Les régles de syntaxe précisé pour la commande cd s'appliquent aussi pour ce cas. Si vous êtes déjà dans le répertoire source ou de destination, vous n'avez donc pas besoin d'entrer un nom de dossier. Pour copier un fichier il suffit juste d'entrer le nom du dosier à la fin du répertoire. Ainsi la copie du fichier image.jpg du dossier1 au dossier3 en le renommant image3.jpg donne (si l'on est dans le dossier 2) `cp dossier1/image.jpg dossier3/image3.jpg`. Si l'on veut prendre tout un dossier `cp -r dossier1 dossier3`. Et si l'on veut tous les fichiers d'un certains types : `cp -r dossier1/*.jpg dossier3`. Vous avez peut être remarqué le `-r` des deux dernières commandes. C'est un paramètre qui permet d'appliquer l'action de manière recursive et donc de compléter l'opération. Les principaux paramètres sont:

- `-i` : Signifie "interactif" et qui va forcer la commmande cp à demander la confirmation du remplacement d'un fichier ou l'abandon de la copie du fichier.
- `-b` : Signifie "backup" et force la commande de créer une copie de sauvegarde de tous les fichiers qui vont être remplacé lors de la copie.
- `-n` : Va demander à la commande de ne pas copier les fichier qui vont en écraser d'autres.
- `-f` : Est l'exact opposé de `-n` et va justement forcer la commande à écrasé les fichiers péexistants lors de la copie.
- `-r` : Va permettre d'effectuer la question de manière recursive et est donc requise pour la copie de plusieurs fichiers ou dossiers.

`ls`

Cette commande permet de lister le contenu du répertoire actuel. Le paramètre `-l` permet un affichage plus détaillé des informations. Tandis que le paramètre `-h` retourne un texte plus lisible pour l'homme. Lors de l'affichage détaillé, les informations fournies sont, dans l'ordre:

- Le type de l'élément, soit un tiret (-) pour un simple fichier, un "d" pour les dossiers et un "l" pour les raccourcisou liens symboliques.
- Les permissions associés à l'élément. Elles se lisent par bloc de trois, avec le premier bloc éttant les permissions du propriétaire de l'élément, le second, les permissions du groupe auquel appartient l'élément, et le toisième bloc représentant les permissions des autres utilisateurs. Ensuite, chaque caractère de bloc repésente soit une lettre soit un tiret (-). Si la lettre "r" est présente cela signifie que la personne concerné à le droit de lecture, le "w" représente le droit d'écriture, tandis que le x le droit d'éxécution. Pour plus d'inforamtions sur les permissions, lire la partie consacrée à `chmod` et `chown`.
- Le nombre de liens physiques.
- Le propriétaire et le groupe de l'élément.
- La taille de l'élément en octet, ou de manière classique (KO, MO, GO) si l'option `-h` est activée.
- L'horodatage de la dernière modification.
- Pour finir, le nom du fichier.

`mkdir`

La commande `mkdir` permet la création d'un dossier. Le paramètre le plus utile est `-m` qui permet de définir les permissions du dossie à sa création. Comme argument du paramètre, il faut entrer un nombre à trois chiffre définissant les permissions. Plus d'informations sur ce nombre dans la partie consacré à `chmod` et `chown`.

`mv`

L'utilisation de la commande `mv` est très similaire à `cp`, avec les mêmes paramètres. Cette commande diffère de `cp`, par le seul fait qu'elle supprime le dossier/fichier d'origine, car elle déplace l'élément. L'autre utilité de la commande `mv` est de permettre de renommer un fichier. Ainsi un fichier nommé image.jpg sera renommé image2.jpg par la commande `mv image.jpg image2.jpg`.

`nano`

La commande `nano` est très utile si vous tentez de configurer, par exemple, un Raspberry Pi en mode "headless" (sans clavier, souris et écran) via le ssh, ou si votre Raspberry est sous Raspbian Lite (pas d'interface graphique). Dans ces deux cas, il est impossible de modifier les fichier en les ouvrants dans votre éditeur texte. Il vous reste alors deux options, la commande `echo` (pas traité ici), ou la commmande `nano`. Ainsi, il vous suffit d'écrire `nano chemindufichier/NOM_DU_FICHIER.extension` pour l'ouvrir dans le micro éditeur de texte intégré dans votre terminal. Une fois les modifications effectuées, la commande `ctrl + o` permet de sauvegarder le fichier, tandis que la commande `ctrl + x` permet de quitter l'éditeur nano et de retourner au terminal lui-même.

`pwd`

La commande `pwd` est utile une fois que l'on est perdu (en terme de répertoire, elle ne donne pas les réponses à toutes vos questions), car elle affiche le chemin exacte du répertoire actuel.

`rm`

La commande `rm` sert à supprimer des fichiers. Elle peut de nouveau être utilisées avec les paramètres `-r` (recursif) et `-f` (force). Pour supprimer tous les fichiers d'un certain type: `rm *.extension`, tandis que `rm -rf mondossier` permet la suppression d'un dossier complet et de son contenu. Enfin il est aussi possible de supprimer deux (ou plus) fichiers en stipulant leurs deux (ou plus) noms: `rm fichier1.extension fichier2.extension`.

`su` et `sudo`

La commande `su` permet de devenir l'utilisateur Root, l'utilisateur qui a tout les droits sur tout les fichiers et peut éxecuter toutes les commandes. Quant à elle, la commande `sudo` signifie Subistute User DO (faire en se substituant à l'utilisateur) et permet de devenir temporairement (le temps d'éxcution d'une commande) l'utilisateur Root. Cependant selon le système d'exploitation, il est impossible d'éxecuter la commande `su` sans être l'utilisateur Root. À ce moment il suffit d'entrer `sudo su`. Une fois devenu l'utilisateur Root, il peut être utile de redevenir sois-même (un simple mortel). Pour cela, entrez `exit`.

`chmod` et `chown`

Les deux commandes permettent de gérer les permissions attribuées au fichiers ou dossiers. Plus précisément, `chown` change le propriétaire d'un fichier ou d'un répertoire. Pour ce faire, il suffit d'entrer la commande suivante: `chown propriétaire:groupe le_fichier_cible`. La commande `chmod` est plus complexe, en effet, elle fournit deux types de syntaxe pour la définission des permissions. La première, la plus logique, change les permissions de manière relatives. C'est à dire que soit vous ajoutez des permissions, soit vous en retirez. La syntaxe et la suivante `chmod QUI+/-permission nomdufichier`, où il faut remplacer "qui" par:

- "a" si vous voulez modifier les permissions pour tout le monde
- "u" si vous voulez modifier les permissions pour le propriétaire
- "g" si vous voulez modifier les permissions pour le groupe auxquel appartient le fichier
- "o" si vous voulez modifier les permissions pour les autres utilisateurs

Ensuite, vient soit le signe plus (+) pour ajouter des permissions, soit le signe moins(-) pour en retirer. Puis les permissions à ajouter ou retirer:

- "r" (read), la permission de lecture
- "w" (write), la permission d'écriture
- "x", la permission d'éxecuter

Par exemple, pour attribuer la permissions d'éxecution à tout les utilisateurs, la commande requise est `chmod a+x nomdufichier`.

`nomdufichier`.

Passons maintenant à la deuxième méthode, qui défini complétement les permissions. Cette méthode s'appuie sur le nombre à trois chifffres évoqué dans la partie traitant de la commande `mkdir`. Le premier chiffre gère les permissions du propriétaire, le deuxième celles du groupe, et le troisième les permissions des autres uilisateurs. Les chiffres quant à eux représentent les ppermissions selon le tableau suivant :

Permissions	Valeur littéral	Valeur numérique
Aucune Permission	---	0
Éxecution seulement	--x	1
Écriture seulement	-w-	2
Éxecution et écriture	-wx	3
Lecture seulement	r--	4
Lecture et éxecution	r-x	5
Lecture et écriture	rw-	6
Toutes les permissions (lecture, écriture, éxecution)	rwx	7

Ainsi la commande `chmod 740 nomdufichier` donne tout les droits au propriétaire, le droit de lecture uniquement aux membres du groupe, et aucun droit sur ce fihcier aux autres utilisateurs.

`shutdown`

La commande `shutdown` permet d'arréter le système lors d'absence d'interfaces graphiques, et si la machine n'a pas de bouton d'allumage/extinction (par exemple le Raspberry Pi). Les deux paramètres importants sont `-r` qui permet de redémarrer le système (la commande équivalente est `reboot`) et `-P` qui permet de demander une déconnection de l'alimentation. Pour définir dans combien de temps la commande doit être éxecutée, il suffit de mettre le nombre de minute en argument (`shutown -r +5` pour redémarrer dans 5 min), ou l'argument `now` pour le faire immédiatement. Par défaut, (en tout cas sur le Raspebrry Pi), le temps d'attente est de 1 minute. Enfin, si vous voulez annuler une extinction planifiée, la commande `shutdown -c` s'en chargera.