TD N°1
Systèmes de gestion de fichiers
(Point de vue utilisateur)

Le système de gestion de fichiers

Le système de fichiers racine (root file system), soit le système de fichiers primaire est associé au répertoire le plus haut **/** :
/bin commandes binaires utilisateur essentielles (pour tous les utilisateurs)
/boot fichiers statiques du chargeur de lancement
/dev fichiers de périphériques
/etc configuration système spécifique à la machine
/home répertoires personnels des utilisateurs (optionnel)
/lib bibliothèques partagées essentielles et modules du noyau
/mnt point de montage pour les systèmes de fichiers montés temporairement
/proc système de fichiers virtuel d'information du noyau et des processus
/root répertoire personnel de root (optionnel)
/sbin binaires système (binaires auparavant mis dans /etc)
/sys état des périphériques (model device) et sous-systèmes (subsystems)
/tmp fichiers temporaires

Partie I : Commandes de bases Linux

Le but de cette partie est la prise en main des commandes de base de l'environnement Linux.

1. Introduction

- Qu'est-ce que le **shell** ?

C'est l'interpréteur de commandes (l'interface) entre l'utilisateur et le système d'exploitation, d'où son nom anglais «shell», qui signifie «coquille».

Le shell est ainsi chargé de faire l'intermédiaire entre le système d'exploitation et l'utilisateur grâce aux lignes de commandes saisies par ce dernier. Son rôle consiste ainsi à lire la ligne de commande, interpréter sa signification, exécuter la commande, puis retourner le résultat sur les sorties.

Il existe plusieurs types de shells, les plus connus depuis Unix ayant une version améliorée sous Linux. Le fichier /etc/shells contient une liste de tous les shells disponibles :

/bin/**ash**
/bin/**bash**
/bin/**bash1**
/bin/**csh**
/bin/false
/bin/passwd
/bin/**sh**
/bin/**tcsh**
/usr/bin/**csh**
/usr/bin/**ksh**
/usr/bin/**tcsh**
/usr/bin/**zsh**

Les plus connus sont bash (version améliorée du shell Bourne sous Unix), ksh (version améliorée du shell Korn sous Unix) et tcsh (version améliorée du shell C sous Unix). La commande help affiche la liste des commandes internes du shell. Par défaut, c'est le shell Bash qui est installé avec Linux. C'est aussi le plus puissant et le plus utilisé, c'est pourquoi c'est celui-ci qui sera utilisé dans les sections suivantes.

Chaque utilisateur possède un shell par défaut, qui sera lancé à l'ouverture d'une invite de commande. Le shell par défaut est précisé dans le fichier de configuration **/etc/passwd** dans le dernier champ de la ligne correspondant à l'utilisateur. Il est possible de changer de shell dans une session en exécutant tout simplement le fichier exécutable correspondant, par exemple : **/bin/bash**

2. Commandes pour débuter

Avant de commencer, il faut savoir que Linux est **sensible à la casse** (*case sensitive* en anglais), c'est à dire qu'il distingue les majuscules des minuscules. Ainsi, pour créer un répertoire, la commande est 'mkdir', ce n'est pas la peine d'essayer MKDIR ou mKdiR, cela ne fonctionnera pas. De même, les noms de fichiers et de répertoires sont également sensibles à la casse.
De plus, sous Unix, les chemins sont séparés par des slash : /etc//init/xfs mais jamais etc\init\xfs.

Répertoires spéciaux :
. représente le répertoire courant,
.. représente le répertoire parent
~ représente le répertoire maison (home) de l'utilisateur

Fichiers cachés :
sous Unix, les fichiers cachés commencent par un point. Par exemple, ~/.bashrc est un fichier caché, dans le répertoire maison de l'utilisateur, qui contient la configuration de son shell.

Jokers : ? *et* *
Les caractères ? et * dans les noms de fichiers et de répertoires permettent de représenter des caractères quelconques. '?' représente un seul caractère, tandis que '*' en représente un nombre quelconque.

Par exemple "*.jpg" représente tous les fichiers se terminant par jpg ; "*toto*" tous les fichiers contenant "toto".

Il faut également savoir que c'est le shell qui interprète ces caractères avant de transmettre la ligne de commande. Par exemple, si vous tapez : rm f *.tmp, le shell transformera cette ligne de commande en : rm truc1.tmp truc2.tmp truc3.tmp.

3. Commandes

Une commande est l'exécution d'un programme dans l'interprète (**Shell**). Elle prend en entrée des options et/ou des paramètres. Elle peut renvoyer de l'information à l'écran ou dans un fichier, modifier un fichier, ou produire un message d'erreur.

Une description de toutes les commandes est disponible avec la commande **man** ou **help**. N'hésitez pas à l'utiliser.

voici les commandes de base sous Linux :

Commandes linux	équivalent MsDos	à quoi ça sert	Exemples :

`cd`	`cd`	change le répertoire courant.	`cd ..`
			- va dans le répertoire parent du répertoire courant
			
			`cd /home/user/.nsmail`
			- va dans le répertoire désigné

`ls`	`dir`	affiche le contenu d'un répertoire	`ls`
			- affiche le contenu du répertoire courant
			
			`ls -l`
			- affiche le contenu du répertoire courant de manière détaillée
			
			`ls -a /home/user`
			- affiche le contenu du répertoire désigné (ainsi que les fichiers cachés)

`cp`	`copy`
	`xcopy`
		copie un ou plusieurs fichiers	`cp toto /tmp`
			- copie le fichier toto dans le répertoire `/tmp`
			
			`cp toto titi`
			- copie le fichier `toto` sur le fichier `titi`
			
			`cp -R /home/user /tmp/bak`
			- copie le répertoire `/home/user` ainsi que tout ce qu'il contient dans `/tmp/bak`

`rm`	`del`	efface un ou plusieurs fichiers	`rm toto titi`
			- efface les fichiers `toto` et `titi`
			
			`rm -f toto titi`
			- efface les fichiers `toto` et `titi` sans demander confirmation

`rm -rf`	`deltree`	efface un répertoire et son contenu	`rm -rf /tmp/*` - efface (sans demander de confirmation) tous les fichiers et répertoire de `/tmp`
`mkdir`	`md`	crée un répertoire	`mkdir /home/user/mes documents` - crée le répertoire `"mes documents"` dans le sous répertoire `/home/user`
`rmdir`	`rm`	efface un répertoire s'il est vide	`rmdir /home/user/.nsmail` - efface le répertoire `.nsmail` de `/home/user` si celui-ci est vide

`mv`	`ren`
	`move`
		déplace ou renomme un ou des fichiers	`mv tata titi`
			- renomme tata en titi
			
			`mv * *.bak`
			- ne fonctionne pas !!!!
			
			`mv * /tmp/bak`
			- déplace tous les fichiers du répertoire courant vers le répertoire `/tmp/bak`

chown	pas d'équivalent	modifie le propriétaire d'un fichier	`chown user unfichier` rend `user` propriétaire de `unfichier`.
chgrp	pas d'équivalent	modifie le groupe propriétaire d'un fichier	`chgrp -R nobody /home/httpd` - rend le groupe : `nobody` (un groupe ayant très peu de droit sur un système linux) propriétaire de `/home/httpd` ainsi que tout les fichiers qu'il contient (`-R`)
ln -s	pas d'équivalent	crée un lien vers un fichier	`ln -s /dev/fd0 /dev/disquette` crée un lien vers `/dev/fd0` (le lecteur de disquette) nommé `/dev/disquette`. La manipulation de `/dev/fd0` et `/dev/disquette` (sauf l'effacement).
grep	pas d'équivalent	recherche une chaine dans un fichier (en fait recherche une expression régulière dans plusieurs fichiers)	`grep chaine *.txt` - recherche la chaine '`chaine`' dans tous les fichier se terminant par `.txt`.
which	pas d'équivalent	trouve le répertoire dans lequel se trouve une commande	`which emacs` - retourne le nom du répertoire dans lequel se trouve la commande `emacs`.
cat	**type**	affiche un fichier à l'écran	`cat ~/.bashrc` - affiche le contenu du fichier `~/.bashrc`

mv	**ren**
	move
		déplace ou renomme un ou des fichiers	`mv tata titi`
			- renomme tata en titi
			
			`mv * *.bak`
			- ne fonctionne pas !!!!
			
			`mv * /tmp/bak`
			- déplace tous les fichiers du répertoire courant vers le répertoire `/tmp/bak`

find	**dir -s**	trouve un fichier répondant à certains critères	`find /home -name "*bash*"` - trouve tous les fichiers contenant le mot `bash` dans leur nom se trouvant dans le répertoire `/home`
locate	**dir -s**	trouve un fichier d'après son nom	`locate bash` - trouve tous les fichiers contenant le mot `bash` dans leur nom complet (avec le répertoire) : à la différence de `find`, locate trouve ses informations dans une base de donnée créée par `updatedb`
man	**help**	affiche l'aide concernant une commande particulière	`man ls` - affiche l'aide (page de `man`uel) de la commnade `ls`. On quitte man en appuyant sur la touche '`q`'

chmod	pas d'équivalent	modifie les permissions d'un fichier	`chmod o+r /home/user`
			- autorise les autres (`o=other`) (ie: ceux qui ne sont ni le propriétaire, ni membre du groupe propriétaire) à lire (`r=read`) le répertoire `/home/user`
			
			`chmod a+rw /home/user/unfichier`
			- autorise tout le monde (`a=all`) à lire et écrire (`w=write`) dans le fichier `/home/user/unfichier`

Aliases

Plutôt que de taper de longues commandes, ou bien parce que vous préférez vous rappeler d'un nom plutôt que du vrai nom Unix, vous pouvez définir des aliases. Pour ce faire, utilisez la commande **alias** comme suit :

Si votre shell est bash ou sh ou ash (par défaut) :
alias md=mkdir
alias ls='ls --color'
alias eclip2='telnet eclip2.ec-lille.fr'

Si votre shell est tcsh ou csh (par défaut) :
alias md mkdir
alias ls 'ls --color'
alias eclip2 'telnet eclip2.ec-lille.fr'

Ainsi pourrez-vous taper md au lieu de mkdir, et eclip2 pour vous connecter à cette machine via telnet ; la commande ls affichera une sortie en couleurs...

Partie II **: le système de fichier UNIX**

La commande ls
Cette commande est omniprésente, aussi il est bon d'en présenter les basiques.

Afficher le listing page par page : ls | less (less est une version améliorée de more)
Afficher le listing en couleurs : ls --color
Afficher aussi les fichiers cachés (commençant par un point) : ls -a
Mettre un '/' après les noms de répertoires : ls -p
Afficher le listing détaillé : ls -l

Tri sur la date
Pour afficher les fichiers d'un répertoire en triant sur la date de mise à jour des fichiers
Afficher les fichiers les plus récents en premier : ls -t
Afficher les fichiers les plus vieux en premier : ls -rt
Mixer avec l'option "l" afin d'afficher le listing détaillé : ls -rtl ou ls -tl

bien sûr, toutes ces options sont mixables, ainsi "ls -altp" affiche tous les fichiers, de façon détaillée, dans l'ordre chronologique, an ajoutant '/' après chaque nom de répertoire.
Exemple de listing

```
[jice@taz jice]$ls -alp
total 144
-rw-r--r--   1 jice  users     24 Aug  2 21:37 .bash_logout
-rw-r--r--   1 jice  users    230 Aug  2 21:37 .bash_profile
-rw-r--r--   1 jice  users    467 Aug  2 21:37 .bashrc
-rw-r--r--   1 jice  users   1452 Aug  2 21:37 .kderc
drwxr--r--  12 jice  users   1024 Aug  2 21:37 .kde/
drwxr--r--   2 jice  users   1024 Aug  2 21:37 Desktop/
-rw-r-----   1 jice  users   1728 Aug  2 21:37 adresses.txt
-rw-------   1 jice  users    144 Aug  2 21:37 motsdepasse.txt
lrwxrwxrwx   1 jice  users     14 Aug  2 21:37 linux -> /usr/src/linux
```

Explication :
La première ligne "total 144" est l'espace disque utilisé par l'ensemble des fichiers du répertoire.

1. La première colonne -rw-r--r--représente les permissions associées au fichier. le premier caractère est un **tiret** pour un fichier, un **d** pour un répertoire, un **l** pour un lien, etc.
   ensuite, on a trois groupes de trois caractères : rw- ou r-- ou rwx ou...
   Le premier groupe représente les permissions associées à **l'utilisateur** (ici, jice), le deuxième celles associées à son **groupe** (ici : users), enfin le dernier est les permissions que **tout le monde** a sur ces fichiers.

   r signifie : possibilité de lire ce fichier / dans ce répertoire,
   w signifie : possibilité d'écrire dans ce fichier / répertoire,
   x signifie : possibilité d'exécuter ce fichier / d'aller dans ce répertoire.

2. nombre d'inodes (partie élémentaire de système de fichiers) qui pointent vers le fichier/répertoire (généralement 1 pour un fichier, 2+le nombre de sous-répertoires pour un répertoire).
3. utilisateur à qui appartient le fichier (jice)

4. groupe auquel le fichier appartient (users)
5. taille en octets
6. date et heure de modification
7. nom du fichier/répertoire.

Exercice :
Dans les systèmes d'exploitation dérives d'Unix, le codage des droits se fait sur 9 bits groupes par 3 bits. Ces droits sont codes en un entier. Pour ce faire, on convient de la correspondance : r = 4; w = 2 et x = 1.
Ainsi, les droits rw-, correspondent à l'entier (r=)4+(w=)2=6. Donc rw-rw-rw- correspond à l'entier 666.

Questions :
1). A quels droits correspondent les entiers 751; 521; 214 et 150 ?

2). Par quels entiers sont codes les droits rw-r- -r- - et rwxr-xr-x ?

4. Editeurs de texte

Un éditeur de texte permet de rentrer du texte dans un fichier afin de le conserver.

L'Éditeur vi >>

vi (prononcez vie-aïe ou [vi:ai]) est l'éditeur de texte de base sous Linux, vous risquez bien d'avoir à vous en servir au plus mauvais moment, c'est à dire lorsque plus rien d'autre ne fonctionne.

Lancer VI

Si vous tentez d'ouvrir un fichier inexistant, vi créera ce fichier. Vous pouvez donc créér un nouveau fichier simplement en tapant [vi nom_du_fichier].

Les modes de VI

vi possède deux modes : le mode "**Insert**" et le mode **normal**. En mode normal vous en pouvez pas insérer de texte dans le fichier, mais les touches du clavier sont autant de touches de commandes. En mode Insert, les touches de commandes (notamment les lettres !) se transforment en vraies lettres que vous pouvez insérer dans le fichier.

Insérer du texte

Lorsque vi s'ouvre, il est en mode normal. Pour passer en mode Insert :
tapez [i] ou [Insert] pour insérer du texte à l'endroit où se trouve le curseur,
tapez [A] pour ajouter du texte à la fin d'une ligne.
En mode Insert, vous pouvez taper du texte, effacer avec la touche [Suppr] ou [Bkspace]. Pour quitter le mode Insert, tapez [Esc].

Remarque : à la suite de votre fichier, vous voyez des lignes vides commençant par le caractère '~'. C'est normal : cela signifie juste que ces lignes sont vides, et les caractères '~' ne seront bien sûr pas enregistrés dans votre fichier.

Les commandes
Après avoir quitté le mode Insert, ou avant d'y être entré, les touches du clavier correspondent à des commandes. Voici ci-dessous les commandes de base qui vous permettront de vous y retrouver :
:q! [Entrée] pour quitter sans sauver,
:w [Entrée] pour enregistrer,
:wq [Entrée] pour enregistrer et quitter,
x efface le caractère qui se trouve sous le curseur,
dd efface la ligne sur laquelle se trouve le curseur,
:u[Entrée] permet d'annuler (ou :undo).

L'Éditeur emacs >>
Emacs est un éditeur qui peut tout faire (mettre en couleur vos sources, gérer vos mails, browser internet, lancer des commandes) et même l'édition de fichier texte.

L'écran d'emacs
L'écran d'emacs (que ce soit un terminal ou une fenêtre) se divise en (au moins) quatre parties :
la première ligne de l'écran qui constitue un menu. Celui n'est utile que sous X. (vous pouvez quand même l'appeler en mode terminal par F10, mais son fonctionnement est loin d'être intuitif).
la dernière ligne de l'écran appelée mini-buffer dans laquelle on tape des commandes
le reste de l'écran qui présente le texte en train d'être édité. Cette zone peut elle-même être divisée en plusieurs zones. Chacune des zones est suivie d'une ligne présentant les caractéristique de ce qui est présent dans cette zone.

La commande Cat
La commande **cat** constitue un éditeur (très) simplifié. Elle permet également d'afficher le contenu d'un fichier entier à l'écran.

cat > fich1 (Enter)
Entrer le texte à stocker dans le fichier (Enter)
CTRL D

Redirections
Linux, comme tout système de type Unix, possède des mécanismes permettant de rediriger les entrées-sorties standards vers des fichiers.
Ainsi, l'utilisation du caractère «>» permet de rediriger la sortie standard d'une commande située à gauche vers le fichier situé à droite :

ls -al /home/hk/ > toto.txt
echo "Toto" > /etc/monfichierdeconfiguration

La commande suivante est équivalente à une copie de fichiers :

cat toto > toto2

La redirection «>» a pour but de créer un nouveau fichier. Ainsi, si un fichier du même nom existait, celui-ci sera écrasé. La commande suivante crée tout simplement un fichier vide :
> fichier

L'emploi d'un double caractère «>>» permet de concaténer la sortie standard vers le fichier, c'est-à-dire ajouter la sortie à la suite du fichier, sans l'écraser.
Fich1.txt >> fich2.txt

De manière analogue, le caractère «<» indique une redirection de l'entrée standard. La commande suivante envoie le contenu du fichier toto.txt en entrée de la commande cat, dont le seul but est d'afficher le contenu sur la sortie standard (exemple inutile mais formateur) :
cat < toto.txt

Enfin l'emploi de la redirection «<<» permet de lire sur l'entrée standard jusqu'à ce que la chaîne située à droite soit rencontrée. Ainsi, l'exemple suivant va lire l'entrée standard jusqu'à ce que le mot STOP soit rencontré, puis va afficher le résultat :
cat << STOP

Les messages d'erreur peuvent être dirigés séparément dans un fichier avec 2> :
startx > startx.log 2> startx.err

ou dirigés vers le même fichier que les messages normaux :
startx > startx.log 2>&1

5. Liens physiques et liens symboliques

Les liens physiques
Le système identifie les fichiers (physiquement) par un identificateur unique qui s'appelle le numéro d' i-noeud.

L'i-noeud est en fait un numéro unique qui identifie le fichier :
Par exemple, pour le fichier /droopy/ma.conf si vous faites :
ls -i /droopy/ma.conf
vous obtiendrez son numéro i-noeud.
Pour créer un autre lien, on utilise la commande ln :
ln /droopy/ma.conf /droopy/conf
Vérification:
ls -i /droopy/ma.conf /droopy/conf
Nous obtenons le même numéro
La commande ls -l indique le nombre de liens que comporte un fichier : C'est le chiffre venant après les permissions.
Les liens symboliques
Cette sorte de lien permet de donner un autre nom au fichier, mais n'utilise pas l'i-noeud physique du fichier.

Pour créer un lien symbolique, il suffit de passer l'option **-s** à la commande **ln** :
ln -s /droopy/ma.conf /droopy/config
Cela va créer un fichier /droopy/config avec un autre i-noeud :
Vérifiez avec :
ls -i /droopy/ma.conf /droopy/config
Nous avons dans ce cas 2 numéros différents.
ls -l /droopy
Indique que config pointe (→) bien sur ma.conf.

Avec le lien symbolique, les permissions de /droopy/config seront les mêmes que pour /droopy/ma.conf.
Il est donc facile d'identifier le lien symbolique d'un fichier avec la commande ls -l, alors qu'il n'en va pas de même pour un lien physique.

6. Compression et archivage de fichiers : Gzip, Zip, et Tar

Compression avec Gzip et Zip

Les fichiers comprimés utilisent moins d'espace disque et se téléchargent plus rapidement que les grands fichiers non comprimés. Vous pouvez comprimer les fichiers Linux à l'aide de l'instrument de compression open-source Gzip ou Zip, qui est reconnu par la plupart des systèmes d'exploitation.

Par convention, les fichiers comprimés se voient attribuer l'extension .gz. La commande Gzip crée un fichier comprimé terminant par .gz; Gunzip extrait les fichiers comprimés et efface le fichier .gz.

Pour comprimer un fichier, entrez la commande suivante à l'invite du shell :
gzip filename.ext

Le fichier sera comprimé et sauvegardé comme filename.ext.gz.

Pour décompresser un fichier comprimé, tapez :
gunzip filename.ext.gz

Le filename.ext.gz est effacé et remplacé par filename.ext.

Si vous échangez des fichiers avec des utilisateurs non LINUX, vous devriez utiliser zip pour éviter les problèmes de compatibilité. Linux peut facilement ouvrir des fichiers zip ou gzip, mais les systèmes d'exploitation non-Linux pourraient avoir des problèmes avec les gzip.
Pour comprimer un fichier à l'aide de zip, entrez ceci :
zip -r filename.zip files

Dans cet exemple, filename représente le fichier que vous créez, et files représente les fichiers que vous voulez placer dans le nouveau fichier :

Pour extraire le contenu d'un fichier zip, entrez :
unzip filename.zip

Vous pouvez compresser plusieurs fichiers en même temps avec zip ou gzip. Enumérez les fichiers en les séparant par un espace :
gzip filename.gz file1 file2 file3 /user/work/school

La commande ci-dessus compresse les file1, file2, file3, et le contenu du répertoire /user/work/school pour les placer dans filename.gz.

Archiver avec Tar
Les fichiers **tar** placent plusieurs fichiers ou le contenu d'un répertoire ou de plusieurs répertoires dans un seul fichier. Il s'agit d'une bonne manière de créer des sauvegardes et des archives. Généralement, les fichiers tar terminent par l'extension .tar.

Pour créer un fichier tar, tapez :
tar -cvf filename.tar files/directories

Dans cet exemple, filename.tar représente le fichier que vous créez et files/directories représente les fichiers ou répertoires que vous voulez placer dans le nouveau fichier.

Vous pouvez utiliser des noms d'accès absolus ou relatifs pour ces fichiers et répertoires. Séparez les noms de fichiers et de répertoires par un espace.

La saisie suivante créera un fichier tar en utilisant un nom d'accès absolu :
tar -cvf foo.tar /home/mine/work /home/mine/school

La commande ci-dessus placera tous les fichiers dans les sous-répertoires /work et /school dans un nouveau fichier appelé foo.tar dans le répertoire dans lequel vous travaillez actuellement.

Commande tar -cvf foo.tar file1.txt file2.txt file3.txt place file1.txt, file2.txt et file3.txt dans un nouveau fichier appelé foo.tar.

Pour afficher la liste du contenu d'un fichier tar, entrez :
tar -tvf foo.tar

Pour extraire le contenu d'un fichier tar, entrez :
tar -xvf foo.tar

Cette commande n'élimine pas le fichier .tar, mais elle place des copies du contenu de .tar dans le répertoire dans lequel vous travaillez actuellement.
La commande tar ne compresse pas automatiquement les fichiers. Vous pouvez compresser les fichiers tar avec :
tar -czvf foo.tar

Les fichiers tar compressés se voient attribuer l'extension .tgz et sont comprimés avec gzip.

Pour décompresser un fichier tar, entrez :
tar -xzvf foo.tgz