1- Introduction à Linux

1.1- Historique

Linus B.Thorvald est l'inventeur de ce système d'exploitation entièrement gratuit. Au début des années 90, il voulait mettre au point son propre système d'exploitation pendant ses loisirs. Linus Thorvald avait pour intention de développer une version d'UNIX pouvant être utilisée sur une architecture de type 80386. Le premier clone d'UNIX fonctionnant sur PC a été Minix, écrit par Andrew Tannenbaum, un système d'exploitation minimal pouvant être utilisé sur PC. Linus Thorvald décida donc d'étendre les possibilités de Minix, en créant ce qui allait devenir Linux. Amusées par cette initiative, de nombreuses personnes ont contribuées à aider Linus Thorvald à réaliser ce système, si bien qu'en 1991 une première version du système a vu le jour. Ce n'est qu'en mars 1992 qu'a été diffusée la première version ne comportant quasiment aucun bug.

Avec le nombre croissant de développeurs travaillant sur ce système, celui-ci a rapidement pu intégrer tous les outils présents sous UNIX. De nouveaux outils pour Linux apparaissent désormais à une vitesse vertigineuse.

L'originalité de ce système réside dans le fait que Linux n'a pas été développé dans un but commercial. En effet aucune ligne de code n'a été copiée des systèmes UNIX originaux (en effet Linux s'inspire de nombreuses versions d'UNIX commerciales: BSD UNIX, System V. BSD UNIX). Ainsi, tout le monde, depuis sa création, est libre de l'utiliser mais aussi de l'améliorer.

Bien que Linux ait été initialement conçu pour fonctionner sur plateforme PC, il a désormais été porté (c'est-à-dire adapté) vers de nombreuses autres plates-formes, telles que Macintosh, stations SPARC, stations DEC Alpha, ...

1.2- Composition d'un système LINUX

Lorsque vous vous procurez une **distribution**, vous n'avez pas besoin de savoir **ce qu'elle contient** pour vous servir de Linux. Et bien, sachez que trois composants sont nécessaires pour faire fonctionner Linux et se retrouver dans un environnement graphique:

Linux comprend un **noyau** (Kernel) qui est le centre nerveux du système. Il permet de faire le lien entre les programmes et le matériel et en fait un système multi-tâches grâce au partage des ressources. Chaque noyau possède une version composée de 3 numéros séparés par un point (par exemple 2.2.10). Le deuxième numéro permet de savoir si la version est stable (numéro pair) ou en cours de développement (numéro impair). Le troisième permet de savoir si la version est récente (plus le numéro est grand, plus la version est récente).

Il comprend également un **serveur X** (et non X-Window). C'est une interface graphique qui a été développée au MIT, permettant de créer des applications graphique fonctionnant sur diverses plateformes. X-Window est l'interface graphique des stations UNIX. Elle est aux systèmes Unix ce que Microsoft Windows est aux PC (n'allez surtout pas dire X-windows au risque de vous faire trucider par un fanatique d'UNIX). L'avantage majeur de ce type d'interface est l'utilisation d'une souris pour remplacer certaines commandes. Sous Linux il existe une implémentation du système X-Window appelée **XFree86**, destiné aux systèmes

Unix sur processeurs de type Intel (386-486-Pentium-AMD-Cyrix). XFree86 supporte un nombre très important de cartes vidéo, mais certaines ne sont pas encore supportées. Toutefois avec l'émergence du système Linux, les nouvelles cartes vidéo vont être de plus en plus rapidement supportées!

Pour bénéficier pleinement du système graphique, il faut un **environnement graphique**. On peut citer le plus connu : **KDE**, mais aussi **Gnome** et **GnuStep**. Détaillons un peu ces trois environnements.

- **KDE** (K Desktop Environment) est sans aucun doute l'environnement graphique le plus utilisé (peut-être parce qu'il reprend les caractéristiques graphiques de Windows95-98). Il est simple d'utilisation, bénéficie d'une bonne interface graphique et de la puissance d'un système d'exploitation Unix.
- **Gnome** (Gnu Network Object Model Environment) est également puissant. Il est facile à utiliser et à configurer. Son logo est un pied formant un G.
- **GnuStep** n'en est qu'à ses balbutiements à l'heure actuelle (11/1999) mais pourrait bientôt se retrouver sur la même marche voire au-dessus des deux autres. C'est un environnement totalement orienté objet, il est très stable, convivial et rapide mais réservé pour l'instant aux développeurs et aux initiés.

1.3- L'arborescence du système

Pour assurer la compatibilité et la portabilité, les systèmes Linux respectent l'unique norme FHS (File Hierarchy Standard).

Hiérarchie de base

/	la racine, elle contient les répertoires principaux
/bin	contient des exécutables essentiels au système, employés par tous les utilisateurs (par exemple, les commandes **ls**, **rm**, **cp**, **chmod**, **mount**, **...**)
/boot	contient les fichiers permettant à Linux de démarrer
/dev	contient les points d'entrée des périphériques
/etc	contient les commandes et les fichiers nécessaires à l'administrateur du système (fichiers **passwd**, **group**, **inittab**, **ld.so.conf**, **lilo.conf**, ...)
/etc/X11	contient les fichiers spécifiques à la configuration de X (contient **XF86Config** par exemple)
/etc/opt	contient les fichiers de configuration spécifiques aux applications installés dans /opt
/home	répertoire personnel des utilisateurs
/lib	contient des bibliothèques partagées essentielles au système lors du démarrage
/mnt	contient les points de montage des partitions temporaires (cd-rom, disquette, ...)
/opt	contient des packages d'applications supplémentaires
/root	répertoire de l'administrateur root
/sbin	contient les binaires systèmes essentiels (par exemple la commande **adduser**)

/tmp		contient les fichiers temporaires
/usr		Hiérarchie secondaire
	/usr/X11R6	ce répertoire est réservé au système X version 11 release 6
	/usr/X386	utilisé avant par X version 5, c'est un lien symbolique vers /usr/X11R6
	/usr/bin	contient la majorité des fichiers binaires et commandes utilisateurs
	/usr/include	contient les fichiers d'en-tête pour les programmes C et C++
	/usr/lib	contient la plupart des bibliothèques partagées du système
	/usr/local	contient les données relatives aux programmes installés sur la machine locale par le root
	/usr/local/bin	binaires des programmes locaux
	/usr/local/games	binaires des jeux locaux
	/usr/local/include	fichiers d'en-tête C et C++ locaux
	/usr/local/lib	Bibliothèques partagées locales
	/usr/local/sbin	binaires systèmes locaux
	/usr/local/share	hiérarchie indépendante
	/usr/local/src	fichiers sources locaux
	/usr/sbin	contient les fichiers binaires non essentiels au système réservés à l'administrateur système
	/usr/share	réservé aux données non dépendantes de l'architecture
	/usr/src	contient des fichiers de code source
/var		contient des données variables

2- Distributions Linux

2.1- Définition

Les **distributions** rassemblent les composants d'un système GNU/Linux dans un ensemble cohérent et stable facilitant son installation, utilisation et maintenance. Elles comprennent donc le plus souvent un logiciel d'installation et des outils de configuration. Il existe de nombreuses distributions, chacune ayant ses particularités propres, certaines sont dédiées à un usage spécifique (pare-feu, routeur, grappe de calcul...) d'autres à un matériel spécifique.

2.2- Historique

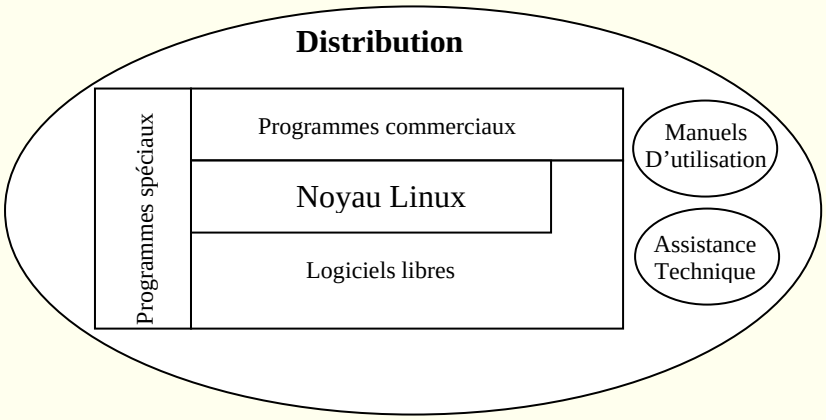
La première distribution est apparue en 1992. Elle se composait de quelques dizaines de disquettes. En raison de la très forte croissance de Linux, une distribution actuelle peut occuper de quelques Méga-octets (pour être installée sur une clé USB par exemple), à plusieurs Giga-octets.

Avant l'existence des distributions, les utilisateurs de GNU/Linux devaient composer eux-mêmes leur système en réunissant tous les éléments nécessaires.

En 1992, Linux (version 0.96) est pleinement fonctionnel. C'est la naissance des premières distributions GNU/Linux : Yggdrasil, MCC Interim Linux, TAMU. Au milieu de l'année, Softlanding Linux System (SLS) est créé: Slackware la plus vieille distribution encore en activité aujourd'hui, est basée sur cette distribution.

2.3- Architecture logicielle d'une distribution

L'intérêt d'une distribution est l'exploitation du concept de couche d'abstraction. Comme on peut le voir sur le schéma les parties qui composent la distribution sont distinctes. On peut donc, par exemple, changer le noyau sans changer les logiciels et donc porter plus facilement la distribution sur une autre architecture matérielle.



2.4- La licence GNU GPL

L'objectif de la licence GNU GPL, selon ses créateurs est de garantir à l'utilisateur les droits suivants (appelés *libertés*) sur un programme informatique :

1. la liberté d'exécuter le logiciel, pour n'importe quel usage ;
2. la liberté d'étudier le fonctionnement d'un programme et de l'adapter à ses besoins, ce qui passe par l'accès aux codes sources ;
3. la liberté de redistribuer des copies ;
4. la liberté d'améliorer le programme et de rendre publiques les modifications afin que l'ensemble de la communauté en bénéficie.

2.5- L'histoire de la GNU GPL

La GNU GPL a été écrite par Richard Stallman pour être utilisée sur les programmes du projet GNU. Elle est basé sur l'assemblage des licences utilisées par GNU Emacs, GNU Debugger (GDB) et la GNU Compiler Collection (GCC). Ces licences contiennent des clauses identiques, mais elles sont spécifiques à chaque programme. Le but de Stallman est de produire une licence unique qui pourra être utilisé pour chaque projet et que cette licence permette au plus grand nombre de projets de partager leur code source. C'est ainsi que naquit la GPL version 1 en janvier 1989.

En 1990, il était devenu évident qu'une licence moins restrictive serait utile pour quelques bibliothèques logicielles. Ainsi, quand la version 2 de la GPL apparut en juin 1991, une nouvelle licence fut créée, la *GNU Library General Public License* (abrégé GNU LGPL ou

LGPL) prenant elle aussi la version 2 pour marquer leur lien de parenté. Les numéros de versions sont devenus différents en 1999 quand la version 2.1 de LGPL est arrivée. La LGPL a changé de nom en même temps afin de mieux réflèter sa place par rapport à l'esprit GNU : elle s'appelle désormais la *GNU Lesser General Public License* (toujours abrégé GNU LGPL ou LGPL).

2.6- Principales distributions

- **RedHat** est une distribution commerciale largement répandue (surtout aux États-Unis). La société RedHat qui la supporte a développé RPM, un gestionnaire de paquets sous licence GPL que d'autres distributions utilisent.
- **Fedora** Le projet Fedora a d'abord eu pour but de développer des paquets RPM complémentaires pour la distribution d'entrée de gamme de Red Hat. Mais au mois de septembre 2003, Red Hat, décidé à se concentrer sur le marché des entreprises, laisse la charge de sa distribution grand public à Fedora.
- **Debian** est une distribution non commerciale régie par le *contrat social Debian*. Elle se distingue par le très grand nombre d'architectures supportées et par son cycle de développement relativement long, gage d'une certaine stabilité. Sa qualité et son sérieux sont unanimement reconnus, mais elle garde l'image d'une distribution réservée aux experts, alors que son ergonomie a bien évolué.
- **SuSE Linux** est la première distribution européenne. Créée en 1993 à Nuremberg, elle a été rachetée par la société Novell en fin 2003. Elle propose 2 distributions majeures : **SuSE Linux Enterprise** pour les besoins d'entreprise (certifications hardware et logicielles nombreuses) et **openSUSE** pour des besoins plus orientées grand public.
- **Mandriva Linux** est une distribution française éditée par la société **Mandriva**. Elle est considérée comme facile d'installation pour les débutants.
- **Slackware** est l'une des plus anciennes distributions existantes. Son installation se faisait initialement à partir d'un jeu d'une trentaine de disquettes et était réputée pour être ardue car très peu automatisée. *Slackware* a été historiquement une des premières permettant de faire tourner Linux *in situ* depuis un CD-ROM, dès 1995.
- **KNOPPIX**, distribution live.

3- L'interpréteur de commandes : SHELL

3.1- Définition

Shell est un mot anglais signifiant « coquille ». En informatique, un **Shell** est une interface qui permet d'accéder aux services proposé par un **noyau** ;

Un **interpréteur de commandes**, est un logiciel faisant partie des composants de base d'un système d'exploitation, le ***command-line interface*** (CLI) (ou '**command prompt***).*

C'est un mode de contrôle d'un ordinateur fonctionnant en **ligne de commande** (commandes tapées au clavier) qui sont transmises au Shell en mode interactif.

Son rôle est de traiter des lignes de commande tapées au clavier. Ces commandes, une fois traitées, interprétées, auront pour effet de réaliser telle ou telle tâche d'administration, ou bien de lancer l'exécution d'un autre logiciel.

3.2- Présentation de quelques shells Linux

- Shell de Stephen R. Bourne (voir (en) Stephen R. Bourne)
  - Bourne shell (sh)
  - Bourne-Again shell (bash)
- csh : C shell
- Shell de David Korn : Korn shell (ksh), voir (en) David Korn
- tcsh
- shell de Kenneth Almquist, utilisé lorsqu'il est nécessaire d'avoir un shell prenant peu de place sur le disque, clone de la variante SVR4 du Bourne shell;
  - Ash (Almquist SHell voir (en) ash (Almquist shell))
  - Dash shell (Debian Almquist SHell)
- Z shell (zsh), reprenant les fonctions les plus pratiques de bash, ksh et tcsh.

3.3- Le Shell Bash

Bash, acronyme de **Bourne-again shell**, est l'interprète de commandes libre du projet GNU. Son nom est un jeu de mots (*Bourne again* / *born again*, il apporte de nombreuses améliorations, provenant notamment du Korn shell et du C shell. Son auteur original est Brian Fox de la Free Software Foundation, relayé plus tard par Chet Ramey. Le Bourne shell original fut écrit par Steve Bourne.

Bash est un logiciel libre publié sous licence GPL. Il est l'interprète par défaut sur de nombreux Unix libres, notamment sur les systèmes GNU/Linux. C'est aussi le shell par défaut de Mac OS X et il a été porté sous Windows par le projet Cygwin.

Bash utilise la bibliothèque readline ce qui lui permet, comme le C shell, de compléter automatiquement les noms de commandes et de fichiers lors d'une frappe sur la touche TAB, ce qui accélère considérablement le travail. Les touches UP et DOWN permettent de naviguer avec facilité dans l'historique des commandes.

La version 3.0 est sortie le 28 juillet 2004.

3.4- La boucle d'interprétation

Le travail de tout interprète de commande peut se résumer à l'algorithme très simple suivant:

TANT QUE l'utilisateur ne ferme pas la session FAIRE

1. émettre un signe d'invite (*prompt*)
2. lire la ligne courante
3. exécuter la commande indiquée sur cette ligne

FIN

La fermeture de session se fera par la commande "exit" ou un "^D" (la marque de fin de fichier sous UNIX: l'utilisateur ferme le flux de lecture de l'interprète, mettant fin à la boucle).

Chaque commande respecte les contraintes suivantes:

1. le *premier* mot de la ligne est interprété comme le *nom* de la commande, les autres mots en sont les paramètres ou options;
2. chaque mot est séparé par un ou plusieurs *caractères de séparation*. Par défaut, ces caractères sont l'espace et la tabulation (modifiable grâce à la variable d'environnement IFS)
3. la fin de la commande est marquée, soit par un ";" (si l'on veut placer plusieurs commandes sur la même ligne), soit par un saut de ligne.

3.5- Évaluation d'une commande

Le premier mot de la ligne est donc considéré par l'interprète comme un nom de commande. Mais il en existe de plusieurs sortes, avec différents types de priorité. Dans l'ordre décroissant de priorité:

1. un *alias*: c'est un mot défini comme synonyme d'un autre (groupe de) mot(s). Dans ce cas, l'interprète commence par remplacer le mot par son équivalent.
2. une commande *interne*: c'est une commande qui a été définie dans l'interprète lui-même, elle ne fait pas l'objet d'un programme séparé. Par exemple, la commande "cd" (*change directory*) est une commande interne du shell.
3. une commande externe, c'est-à-dire un programme (ou un fichier de commande, en général appelé *script*) stocké sur le disque de la machine. Par exemple, "/bin/ls" est la commande qui permet de lister les fichiers contenus dans un répertoire donné.

Dans le cas d'une commande externe, le shell doit retrouver le programme avant de pouvoir l'exécuter. Il est clair qu'une recherche exhaustive, pour chaque commande, dans tout le système de fichiers prendrait beaucoup trop de temps et rendrait le système inutilisable dans la pratique. C'est la raison pour lequel seul un petit ensemble de répertoires est en fait consulté lors de la recherche du programme permettant l'exécution de la commande. Ces répertoires sont définis dans une variable d'environnement appelée "PATH" (chemin). Si la commande n'est pas trouvée dans l'un des répertoires listés dans le PATH, l'exécution se soldera par un message d'erreur du type "command not found".

3.6- Shells et environnements graphiques

L'invite est l'interface la plus simple à réaliser et conserve de nombreux avantages par rapport aux environnements graphiques :

- précision et simplicité d'automatisation des tâches (mode batch) ;
- contrôle à distance ;
- uniformité ;
- stabilité ;
- faible consommation des ressources.

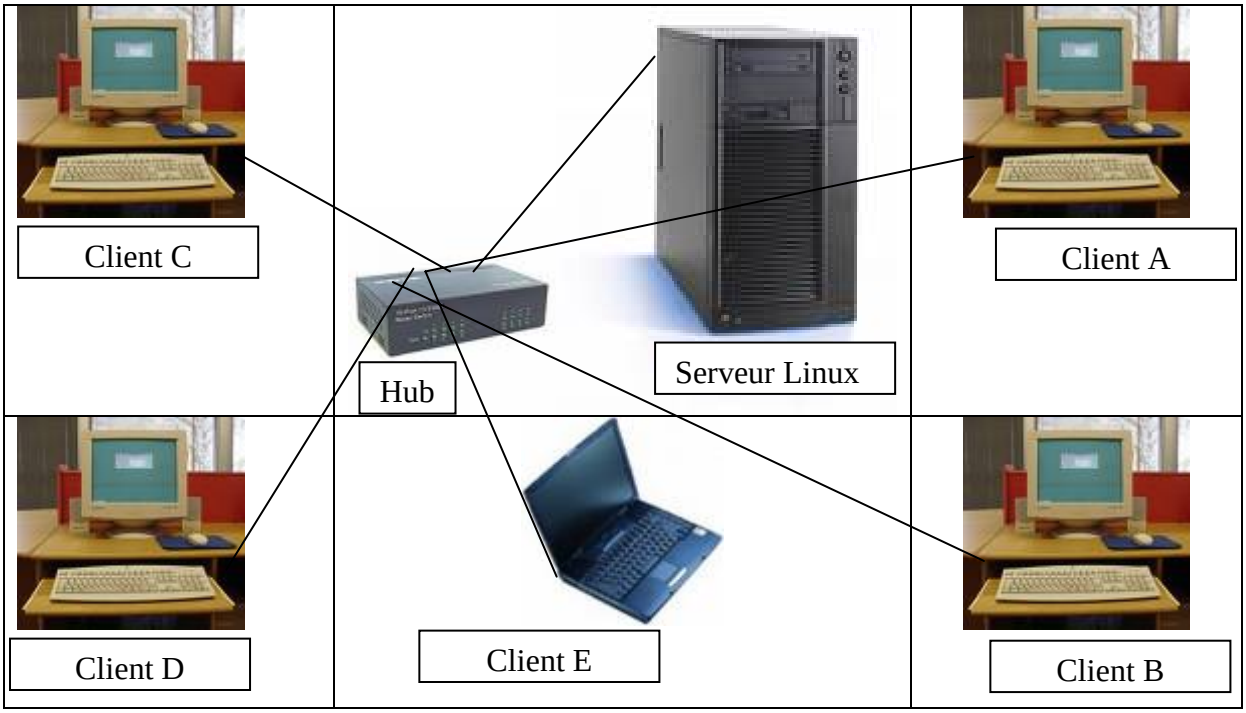
Beaucoup de serveurs ne s'administrent qu'en ligne de commande car il y a peu de raisons d'avoir besoin d'une interface graphique sur un serveur, interface qui pourrait être source de dysfonctionnements et dont la prise en main distante est plus ardue.

Dans l'utilisation bureautique quotidienne, les gestionnaires de fichiers graphiques, et autres menus de lancement d'applications tendent à remplacer le shell, en fournissant une alternative plus conviviale. Néanmoins, le shell reste l'outil le plus polyvalent, encore irremplaçable pour

certaines tâches. Ainsi shell et logiciels utilitaires en mode graphique sont complémentaires dans toute utilisation poussée de l'outil informatique.

4- Environnement de travail adopté en salle de cours

L'environnement de travail est basé sur la simulation d'un environnement terminal/serveur tout en utilisant des machines clientes autonomes et non des terminaux passifs. La communication entre le serveur et les clients se fait à travers le service réseau Telnet



Telnet (**TE***rminal* **NET***work* ou **TEL***ecommunication* **NET***work*, ou encore **TEL***etype* **NET***work)* est un protocole de type client-serveur basé sur TCP. Les clients se connectent généralement sur le port 23 du serveur.

Une des utilisations majeures de la commande `telnet` était de se connecter à des serveurs telnet, qui demandaient un identifiant, puis un mot de passe, et donnaient une ligne de commande sur la machine distante en échange. Pour cela elle nécessitait le lancement d'un démon sur la machine hôte, souvent appelé `telnetd`.

La commande `telnet` reste une commande très pratique pour tester des serveurs. Vu la flexibilité du programme, il est possible d'utiliser la commande `telnet` pour établir une connexion TCP interactive avec d'autres services tel que SMTP ou HTTP.

Cependant le côté basique de Telnet fait que toute communication est transmise en clair sur le réseau, mots de passe compris ce qui représente un défaut de sécurité. Des *sniffer*s comme tcpdump ou Wireshark permettent d'intercepter les communications de la commande `telnet`. Des protocoles chiffrés comme SSH ont été développés pour fournir un accès distant remplaçant Telnet et dont l'interception ne fournit aucune donnée utilisable à un éventuel espion.

- Pour se connecter au serveur, il faut disposer au préalable d'un compte utilisateur sur le serveur Linux.
- Soit user1 ce compte utilisateur. On peut accéder d'une machine cliente (environnement Windows) via l'invite de commande et en tapant :
  telnet Adresse IP du serveur (ou le nom de domaine)
  ex : telnet 192.168.1.111
  ou encore : telnet suivi de open 192.168.1.111
  l'utilisateur est invité à saisir son login et son mot de passe

Vous pouvez également utiliser le service ssh pour se connecter au serveur Linux. Pour se connecter au serveur Linux, il y a l'utilitaire graphique Putty, téléchargeable gratuitement d'Internet, qui est un exécutable qui propose d'ouvrir des sessions texte sur un serveur distant en utilisant différents services, tel que telnet ou ssh.

5- Commandes Linux

La syntaxe générale d'une commande sous linux est :
Nom_de_la_commande [- options] [arguments (paramètres)]
Il est possible d'exécuter des commandes sans options ni arguments, à ce moment le système prend les arguments par défaut.

Commandes usuelles sous Linux

COMMANDE	DESCRIPTION
ls	liste le contenu d'un répertoire
rm	supprime un fichier
cp	copie des fichiers ou répertoires
mv	déplace des fichiers ou répertoires
cd	change de répertoire
cd ..	retourne au répertoire parent
clear	efface tous les lignes du terminal
mkdir	crée un nouveau répertoire (make directory)
rmdir	supprime un répertoire vide (remove directory)
pwd	permet d'afficher le répertoire actif (print working directory)
file	permet de connaître le contenu probable du fichier spécifié
!!	exécute la dernière ligne de commande
man **apropos**	aide sur la commande demandée pour une aide sur la fonction **locate**, on tape : **man locate**
chmod	change l'attribut d'un fichier ***chmod XXX fichier*** XXX= Utilisateur|Groupe|Autres ou X représente un entier 1≤X≤7 Lecture=4, Ecriture=2, Exécution=1

	X=Lecture+Ecriture+Exécution
passwd	change le mot de passe de l'utilisateur
cat	affiche le contenu du fichier spécifié à l'écran Cette commande ne doit être utilisée que pour des fichiers pour lesquels la commande file a retourné : english test, ascii text ou commands text Il ne faut surtout pas utiliser la commande cat sur des fichiers dont le contenu supposé est data ou exécutable
more	affiche le contenu du fichier avec des pauses
bye	déconnexion
type	permet de Exécution le chemin d'un exécutable : ***type nom du programme***
tar ***(sans décompresser)***	permet de connaître le contenu d'un fichier.tar.gz (ou .tgz) sans tout décompresser : ***tar tzf exemple.tar.gz	less***
userdel	permet de supprimer un utilisateur. Suppression totale (répertoire privé, boite aux lettres...) avec : ***userdel -r nom_user***
date	affiche la date et l'heure du système
who	indique les utilisateurs connectés au système
cal	affiche un calendrier

5.1- Exemples

La commande ls

Cette commande affiche la liste des fichiers accessibles. Elle s'utilise en ajoutant une ou plusieurs options et le nom d'un répertoire :**ls [options] [répertoire]**
Il existe de nombreuses options et en particulier : -l, -a, -p et -d. Elles sont combinables : -al, -lad, ...
L'option **-l** permet d'afficher toute une série d'informations sur chaque fichier listé.
L'option **-a** affiche tous les fichiers et les répertoires commencant par un point.
L'option **-d** n'affiche pas le contenu du répertoire mais uniquement le répertoire lui-même.
L'option **-p** permet de rajouter un slash après les noms de répertoires.

On peut également utiliser l'option page par page en tapant : **ls | less**. On peut afficher en couleur avec **ls --color**

On peut également rajouter la commande **ls -lrt** qui permet de visualiser les fichiers les plus récents du répertoire.

La commande cp

Cette commande permet de copier un fichier. Deux formes de copie existent :

cp Fichier1 Fichier2 : copie du Fichier1 sous le nom Fichier2

cp Fichier1 [Fichier2 Fichier3 ...] répertoire : le ou les fichiers sources indiqués sont

copiés dans le répertoire spécifié. Les fichiers sources conservent le nom original.
Si le fichier cible existe déjà, il sera écrasé. Alors faites attention...

5.2- Opérations sur les fichiers et dossiers

Après avoir vu la structure du système de fichiers nous allons voir comment travailler sur les fichiers et les dossiers.

Pour se déplacer d'un dossier à un autre on utilise la commande cd. Lancez une console et lancez une commande pwd vous verrez que vous êtes dans votre dossier personnel (c'est le dossier de démarrage par défaut).

Si vous voulez aller dans le dossier **/etc** tapez la commande cd /etc ; si vous voulez aller dans le dossier perso contenu dans votre dossier personnel alors que vous vous y trouvez tapez cd perso. Pourquoi y a t'il un / dans l'un et pas dans l'autre ? Tout simplement car le premier chemin est absolu et part donc de la racine (/) alors que le deuxième est relatif à votre position actuelle (et n'a donc pas besoin de partir de la racine). Si on devait taper le deuxième chemin en absolu cela donnerait cd /home/user/perso. Si vous vous trouvez dans le dossier **/etc** et que vous voulez vous rendre dans le dossier **/etc/rc.d/init.d** voila ce que cela donnerait en chemin relatif : cd rc.d/init.d.

Voici d'autres commandes basées sur cd :
- cd .. vous permet de remonter dans le dossier parent de celui où vous vous trouvez. Si vous vous trouvez par exemple dans votre dossier personnel (**/home/user**) la commande cd .. vous fera remonter dans le dossier **/home**. Et si vous voulez remonter de deux dossiers il vous suffit de taper cd ../..
- cd sans argument vous permet de revenir dans votre dossier personnel.
- cd / vous permet de remonter à la racine de l'arborescence.

Pour créer un dossier il vous faut utiliser la commande mkdir. Le chemin du ou des dossiers à créer pouvant être absolus ou relatifs. La structure type de la commande est (ce qui est entre crochets est facultatif) :
mkdir dossier1 [dossier2] [dossier3] [dossier4] [etc...]

Si vous voulez créer toute une arborescence, pensez à commencer par créer le ou les dossiers principaux avant de créer leurs fils. Par exemple pour créer une arborescence dans votre dossier personnel alors que vous vous y trouvez tapez :
mkdir documents documents/personnel documents/professionnel documents/personnel/photos documents/personnel/textes etc...

Comme vous le voyez créer toute une arborescence sous Linux est très simple : une seule ligne de commande suffit. C'est l'une des petites choses pour lesquelles passer par la console peut être plus rapide que de passer par le mode graphique.

Pour créer un fichier de type texte vous pouvez utiliser Vi (je détaillerai son fonctionnement détaillé ultérieurement) en tapant vi nom_du_fichier_à_créer.

Pour effacer un fichier il vous faudra être le propriétaire de ce fichier et taper la commande rm nom_du_fichier_à_effacer. Pour effacer un dossier et tout ce qui se trouve dedans (sous-dossiers y compris) il vous faudra utiliser l'option -R ou -r : rm -R nom_du_dossier. Attention rm ne met pas les fichiers dans la corbeille il les supprime définitivement, et sous certaines distributions vous n'aurez pas de garde fou pour vous demander confirmation avant de supprimer le ou les fichiers !

Si vous souhaitez qu'il vous demande confirmation alors il vous faudra utiliser l'option -i : rm -Ri nom_arborescence par exemple. Il va alors vous demander confirmation pour chaque fichier à supprimer (ce qui peut être pénible si vous avez beaucoup de fichiers à supprimer). Si par contre vous avez une distribution qui vous demande systématiquement confirmation et que vous ne voulez justement pas avoir à répondre pour chaque fichier à supprimer, alors vous pouvez utiliser l'option -f : rm -Rf nom_arborescence par exemple. Dans ce cas vous forcez la suppression et il ne vous posera pas de question de confirmation (à utiliser donc avec prudence). L'option -f permet aussi de forcer la suppression d'un fichier en lecture seule.

Nous verrons plus loin la notion de droits d'accès sur les fichiers qui explique pourquoi vous ne pouvez supprimer que les fichiers vous appartenant (c'est une des nombreuses sécurité issues d'Unix).

Pour faire une copie d'un fichier sous un autre nom vous devrez taper la commande : cp ancien_nom nouveau_nom. Vous obtenez alors deux fichiers identiques avec deux noms différents. Pour copier un fichier en gardant son nom mais dans un autre dossier il vous faudra taper la commande suivante : cp fichier dossier_de_destination.

La commande cp permet de copier plusieurs fichiers vers un seul dossier :
cp fichier1 fichier2 fichier3 fichier4 [...] dossier_de_destination

La syntaxe globale de cp est :
cp source_unique_ou_multiple destination_unique

La commande cp permet aussi de copier des arborescences entières grâce à l'option -R :
cp -R dossier1/a dossier2/b

Attention : si le nom de fichier de destination existe déjà cp va l'écraser avec le nouveau fichier sans poser de question. Si vous voulez qu'il vous demande confirmation avant d'écraser un fichier il vous faudra utiliser l'option -i.

Si vous ne souhaitez pas copier le fichier mais le renommer alors il vous faudra utiliser la commande mv : mv ancien_nom_fichier nouveau_nom_fichier. Si vous souhaitez déplacer le fichier dans un autre dossier alors tapez mv nom_fichier dossier_de_destination.

La commande cp consomme des Inodes et donc de l'espace disque. Si vous avez juste besoin de créer un lien vers un fichier et que vous ne souhaitez pas consommer de l'espace disque, vous pouvez utiliser la commande ln : ln fichier1 fichier2. Si vous faites un ls -li dans le dossier où se trouve le fichier vous verrez que celui-ci et son lien ont le même numéro d'Inode. Le lien n'est qu'une ligne de plus dans le "fichier liste" du dossier pointant vers l'Inode du fichier. Donc pas de consommation d'un Inode de plus.

On peut aussi faire un lien symbolique c'est-à-dire un fichier pointant vers un autre fichier grâce à l'option -s : ln -s fichier lien_vers_fichier. L'intérêt de faire un lien symbolique plutôt d'un lien normal est qu'il permet de faire des liens entre différents systèmes Unix, et de maintenir l'arborescence traditionnelle même lorsqu'on déménage certaines commandes. Allez par exemple regarder le dossier **/etc/rc1.d** et vous verrez qu'il pointe en réalité vers **/etc/rc.d/rc1.d**. Le désavantage du lien symbolique est qu'il crée un fichier et consomme donc un Inode, c'est-à-dire de l'espace disque.

Pour vérifier la quantité d'espace disque utilisée tapez la commande du, et si vous souhaitez voir la quantité d'espace libre tapez la commande df. La commande du vous affichera l'espace occupé par chaque dossier ou sous dossier contenu dans le dossier où vous vous trouvez, df lui affichera l'espace libre pour chaque partition. Les options de du et df vous permettront d'afficher ces résultats en octets, en kilo-octets, ou en %. Voyez le man pour les différentes options.

Après avoir vu comment créer, supprimer, copier, déplacer, renommer un fichier, et faire un lien vers un fichier, nous allons voir comment pousser plus loin les capacités du shell. On ne connaît pas toujours le nom exact d'un fichier, d'un dossier ou d'une commande. Si vous connaissez le début du nom vous pouvez utiliser une petite astuce de l'interpréteur de commandes qui marche aussi bien pour les noms de fichiers et de dossiers que pour les noms de commandes. Tapez les lettres dont vous êtes sûr puis appuyez sur la touche de tabulation (celle avec les deux flèches allant en sens contraire), le shell va alors vous proposer tous les noms de fichiers, de dossiers, ou de noms de commandes possibles. Il vous suffit alors de finir le nom et de taper entrée. S'il n'y a qu'un choix possible le shell va reproduire la ligne de commande que vous aviez tapé en complétant le nom incomplet.

Exemple : si vous avez dans votre dossier personnel un sous dossier nommé **acpica-unix-20060210**, que ce dossier est le seul commençant par **acpica** et que vous souhaitez vous déplacer dedans grâce à la commande cd, il vous suffit de taper cd acpica puis d'appuyer sur la touche de tabulation vous aurez alors à la ligne suivante cd acpica-unix-20060210, il vous suffit alors de taper sur Entrée et c'est bon.

Si vous avez d'autres dossiers commençant par **acpica**, le shell va vous faire une liste de tous ces noms et recopier la ligne de commande en poussant le nom recherché jusqu'au maximum. Par exemple si vous avez deux dossiers nommés **acpica-unix-20051204** et **acpica-unix-20060210**, ces deux noms ont comme partie commune **acpica-unix200**, le shell va donc vous lister les deux noms possibles et vous recopier la ligne la ligne suivante : cd acpica-unix200.

Si vous avez trop de fichiers avec une base commune, ou si vous cherchez plusieurs fichiers ou dossier, vous pouvez pousser les possibilités de la commande ls grâce aux caractères génériques. Ceux-ci permettent de simplifier les recherches de fichiers :
- ***** remplace un ou plusieurs caractères
- **?** Remplace un seul caractère et implique forcément sa présence
- **[...]** remplace un caractère par l'un de ceux mis entre crochets, ou l'un de ceux compris dans la fourchette mise entre crochets ([0-9] par exemple)
- **[!...]** exclut les caractères mis entre crochets.

On peut combiner les caractères génériques :
- ls com*[0-9] trouvera tous les fichiers commençant par **com** et suivis ou non d'une chaîne de caractères puis d'un chiffre. Exemple : **combien2** ou **comment3**.

- ls com*[!0-9] trouvera tous les fichiers commençant par **com** et suivis ou non d'une chaîne de caractères mais ne se finissant pas par un chiffre. Exemple : **combien-fichiers** ou **commandeshell**. Par contre des fichiers comme **combien2** ou **comment3** seront ignorés.

Les crochets équivalent à un seul caractère et impliquent sa présence comme pour le point d'interrogation. Si ce caractère n'est pas présent on aura une erreur. Ces caractères génériques marchent avec la plupart des commandes.

Si vous voulez regarder le contenu des dossiers cachés (dont le nom commence par un point) sans que ls ne remonte dans l'arborescence (en fouillant le fichier nommé ..), les caractère génériques vous y aideront : ls .[!.]* vous permettra d'explorer tous les fichiers et dossiers dont le nom commence par un point mais qui ne commencent pas par deux points, excluant de ce fait la remontée dans l'arborescence.

Attention avec les caractères spéciaux vous pouvez faire de gros dégâts surtout si vous êtes logué en tant qu'administrateur. Faites toujours un ls avec les caractères spéciaux que vous voulez utiliser avant de taper la commande voulue, cela vous évitera d'effacer par exemple tout votre système avec un bête rm -R * tapé à partir de la racine en tant que root.

Voyons quelques autres commandes bien pratiques. la commande cat nom_du_fichier vous permet de visualiser le contenu d'un fichier texte à l'écran, mais ce n'est pas très pratique si le fichier est long et prend plus d'un écran de long. Dans ce cas il est plus intéressant d'utiliser more ou less. Si vous souhaitez lire un fichier par tranche de 8 lignes il vous suffit de taper more -8 nom_du_fichier.

Si vous ne souhaitez voir que les 5 premières lignes d'un fichier il vous suffit de taper head -5 nom_du_fichier. Si vous souhaitez voir les 10 dernières lignes d'un fichier vous devrez taper tail -n 10 nom_du_fichier. D'autres options permettent de lire un fichier à partir d'une ligne précise, et les caractères génériques peuvent vous permettre de lire les premières lignes de plusieurs fichiers (head -3 com* vous permettra de visionner les 3 premières lignes de tous les fichiers commençant par **com**).

Pour comparer le contenu de deux fichiers textes vous pouvez utiliser la commande diff : diff fichier1 fichier2. Si les deux fichiers sont identiques vous n'aurez aucun message. S'ils sont différents diff indiquera les lignes en moins et celles en plus. La commande cmp fait plus ou moins pareil mais avec moins d'options. Pour comparer deux arborescences vous pouvez faire un ls -l dossiers | fichier-arbo de chaque arborescence et comparer les fichiers avec la commande diff.

La commande split nom_du_fichier vous permettra de tronçonner un fichier texte en plusieurs fichiers. Par défaut il crée des fichiers nommés **xaa xab xac** etc ... mais on peut spécifier une autre racine que **x** (voir les options dans le man). Après pour recoller les morceaux il vous suffit de faire un cat x?? >> fichier.

Si vous souhaitez chercher une chaîne de caractères dans un ou plusieurs fichiers textes il vous faudra utiliser la commande grep : grep [options] chaîne_cherchée fichier_où_chercher. grep -n qui * va chercher le mot "**qui**" dans tous les fichiers du dossier et affichera devant chaque occurrence le numéro de la ligne où elle se trouve. grep -i qui * va chercher le mot "**qui**" en ignorant la casse (c'est-à-dire en cherchant chaque lettre aussi bien en minuscule qu'en majuscule). L'option -y fait la même chose. Les caractères génériques peuvent être

utilisés mais ils n'ont pas toujours la même valeur. L'étoile fonctionne de la même façon que décrit précédemment, par contre le point d'interrogation est remplacé par un point simple.

Si vous souhaitez trier le contenu d'un fichier il vous faut utiliser la commande sort : sort fichier va trier le contenu du fichier texte par le contenu de la première colonne (par défaut le séparateur entre deux colonnes est l'espace). Si dans votre fichier le séparateur est **:** au lieu de l'espace il vous faut changer le séparateur par défaut de la commande grâce à l'option -t suivie du nouveau séparateur : sort -t: fichier. Pour trier par colonne il vous faut ajouter **+0** après sort pour trier par la première colonne, **+1** pour trier par la deuxième, **+2** pour trier par la troisième, etc ... Si on veut trier sur la 3e colonne puis sur la 2e colonne sur un fichier qui en comporte 4, il vous faut taper sort +2 -3 +1 fichier. Le **-3** sert à exclure la 4e colonne du tri (il faut penser à exclure les colonnes qui suivent celles que vous sélectionnez quand vous commencez par une plus avancée sinon sort en tient compte). Vous pourrez découvrir les multiples autres options de sort grâce aux pages du man.

Si vous avez besoin d'extraire une colonne d'un fichier texte structuré sous forme de tableau vous devrez utiliser la commande cut : cut -d":" -f2 nom_du_fichier vous permettra d'extraire la deuxième colonne du fichier (spécifié par **-f2**) le séparateur entre chaque colonne étant : (spécifié par **-d":"**). Si on voulait extraire la première colonne il faudrait taper **-f1**, et si le séparateur était l'espace il faudrait taper **-d" "** (ne pas oublier les **""** autour du séparateur).

Si vous voulez convertir des caractères dans un fichier il vous faudra utiliser la commande tr : tr 'a-z' 'A-Z' fichier3 vous permettra de transformer tous les **:** du fichier en **;** et d'enregistrer le nouveau texte dans **fichier3**, ce qui peut être très utile pour transformer une base de données d'un format à un autre en changeant tout simplement le séparateur. Le seul format universel pour les bases de données étant le format texte il vous suffit d'exporter votre base au format texte et de modifier les séparateurs grâce à tr, d'enregistrer les changements dans un nouveau fichier grâce aux redirections puis de réimporter le nouveau fichier.

Enfin voici une commande qui peut très vite devenir très complexe et remplacer parfois des lignes entières de script shell. Il s'agit de la commande find. Sa structure globale est la suivante :
find chemin -conditions -action

C'est une commande conditionnelle. Le chemin est récursif par défaut (si on lui met **/** il cherchera partout) et il est obligatoire (. pour le répertoire courant, **/** pour la racine, **/home**, etc...). Les conditions peuvent être très variées : on peut chercher les fichiers en fonction du user ou du groupe propriétaire, en fonction du nom, de la taille, de l'Inode, des dates d'accès et de modification, des droits d'accès, etc... L'action peut être de trois sortes :
- print : affiche à l'écran ce que find a trouvé (à faire avant toute autre action)
- exec : exécute une commande (toutes les commandes sont possibles)
- ok : exécute une commande mais demande confirmation d'abord.

Si on utilise **exec** ou **ok** voila la syntaxe précise :
find chemin -conditions -exec commande {} \;

Les **{}** servent de boucle dans laquelle les noms des fichiers trouvés par find viennent se mettre. Voici quelques exemples de commande avec find :
find . -name "demo*" -print => va chercher puis afficher tous les fichiers du dossier courant et de ses sous-dossiers dont le nom commence par **demo**

find . -name "comb*" -exec rm {} \; => va chercher puis effacer tous les fichiers du dossier courant et de ses sous-dossiers dont le nom commence par **comb**

Les autres arguments conditionnels peuvent être -links n pour chercher les fichiers en fonction du nombre **n** de liens pointant vers les fichiers ; -links +n pour chercher les fichiers qui ont plus de **n** liens pointant vers eux, -links -n pour chercher les fichiers qui ont moins de **n** liens pointant vers eux. L'argument -user nom_du_user permet de chercher les fichiers appartenant à un user particulier. L'argument -mtime n permet de chercher les fichiers qui ont été modifiés il y a n jours. Il y a bien d'autres arguments possibles. Je vous laisse les découvrir grâce au man. Pensez bien à faire d'abord un find suivi d'un print avant de faire le même find avec un exec ou un ok.

Vous pouvez combiner plusieurs conditions :
- condition1 a condition2 : condition 1 ET (AND) condition2
- condition1 o condition2 : condition1 OU (OR) condition2
- ! condition : exclut la condition
- \(condition1 o condition2 o condition3 .... \) : permet de mélanger AND et OR ou d'utiliser un grand nombre de conditions sans que ça finisse en bloubiboulga. Bien penser à laisser un espace à chaque fois entre la parenthèse ou le back-slash (\ ) et la condition proche de celle-ci.

6- Le fichier /etc/passwd

C'est lui qui contient toutes les informations relatives aux utilisateurs (login, mots de passe, ...). Seul le superutilisateur doit pouvoir le modifier. Il faut donc modifier les droits de ce fichier de façon à ce qu'il soit en lecture seul pour les autres utilisateurs.

Si on regarde de plus près la composition de ce fichier, on s'aperçoit qu'il respecte le format suivant :

nom_du_compte : mot_de_passe : numero_utilisateur : numero_de_groupe : commentaire : répertoire : programme_de_demarrage

Sept champs sont explicités séparés par le caractère "**:**" :

- le **nom du compte** de l'utilisateur

- le **mot de passe** de l'utilisateur (codé bien sûr)

- l'**entier** qui identifie **l'utilisateur** pour le système d'exploitation (UID=User ID, identifiant utilisateur)

- l'**entier** qui identifie le **groupe** de l'utilisateur (GID=Group ID, identifiant de groupe)

- le **commentaire** dans lequel on peut retrouver des informations sur l'utilisateur ou simplement son nom réel

- le **répertoire de connexion** qui est celui dans lequel il se trouve après s'être connecté au système

- la **commande** est celle exécutée **après connexion** au système (c'est fréquemment un interpréteur de commandes)

Exemple

Voici un exemple de fichier ***passwd*** :

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14::/var/lib/uucp/taylor_config:/bin/bash
said:x:500:100:Cool.....:/home/said:/bin/bash

- Il faut savoir que les mots de passe situés dans ce fichier sont chiffrés. Ca sert donc à rien de l'éditer et de remplacer le champ ***mot_de_passe*** en tapant directement son mot de passe. Vous n'obtiendriez uniquement que le blocage du compte.

- Lorsqu'un utilisateur se connecte, le programme login compare le mot de passe tapé par l'utilisateur (après l'avoir chiffré) à celui qui est dans le fichier passwd. Si ils sont différents, la connexion ne peut se faire.

- Pour interdire l'utilisation, il suffit de remplacer le mot de passe chiffré par une étoile : "*****".

- On peut également ouvrir les accès à un compte en laissant le champ ***mot_de_passe*** vide. Toute personne voulant se connecter avec ce compte pourra le faire.

- Pour pouvoir modifier le mot de passe d'un compte grâce à la commande ***passwd***, il faut être soit administrateur système, soit propriétaire du compte.

- **UID** : identifiant (unique) de chaque compte utilisateur. Les nombres de 0 à 99 sont fréquemment réservés à des comptes propres à la machine. Les valeurs supérieures à 100 sont elles réservées aux comptes utilisateurs.

- **GID** : identifiant de groupe. Le groupe par défaut (nommé **group**) porte le numéro 50. Cet identifiant est utilisé en relation avec les droits d'accès aux fichiers. Cette question ne vous préoccupera que si votre système comporte plus d'un seul groupe d'utilisateurs. (Il faudra alors se préoccuper du fichier ***/etc/group***.

- On peut à partir du Shell, modifier l'interpréteur de commandes. Pour ceci, on utilise la commande ***chsh*** ou alors ***passwd -s***. Linux cherche alors dans le fichier ***/etc/shells*** le programme que vous avez spécifié. Seules les commandes présentes dans ce fichier seront acceptées et remplaceront la valeur actuelle du champ ***programme_de_demarrage***. Ces restrictions ne s'appliquent pas au compte du superutilisateur.

- Assurez-vous que les droits d'accès du fichier ***/etc/shells*** sont les mêmes que pour le fichier ***/etc/passwd***

- Le superutilisateur ne se nomme pas obligatoirement **root**. Pour le changer, il suffit de remplacer le nom du compte root par celui désiré.

- Un compte privilégié est un compte dont l'identifiant (UID, User ID) vaut zéro.

7- Le fichier /etc/group

Ce fichier contient la liste des utilisateurs appartenant aux différents groupes. En effet, lorsque de nombreux utilisateurs peuvent avoir accès au système, ceux-ci sont fréquemment rassemblés en différents groupes ayant chacun des droits d'accès différents aux fichiers et aux répertoires. On retrouve ainsi ces regroupements dans le fichier /etc/group.

Il se compose de différents champs séparés par "**:**" :

nom_de_groupe : champ_special : numero_de_groupe : membre1, membre2

Le champ spécial est fréquemment vide.

Le numéro de groupe est le numéro qui fait le lien entre les fichiers /etc/group et **/etc/passwd**

Exemple

Voici un exemple de fichier ***/etc/group*** :

root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:
tty:x:5:
disk:x:6:
lp:x:7:
wwwadmin:x:8:
kmem:x:9:
wheel:x:10:
mail:x:12:cyrus
news:x:13:news

- Lorsqu'on utilise la commande **ls** avec l'option -l, le numéro de groupe est affiché avec le numéro de l'utilisateur à qui appartient le fichier (ou le répertoire). Ce numéro unique correspond à un nom de groupe unique (souvent 8 caratères max.).

- Un même utilisateur peut apparaître dans plusieurs groupes. Lorsqu'il se connecte au système, il appartient au groupe spécifié dans le fichier **/etc/passwd** (le champ GID). Il peut en changer à l'aide de la commande **newgrp**. Des droits d'accès aux fichiers sont alors définis.

- Les protections du fichier doivent empêcher sa modification par les utilisateurs non privilégiés.

Manipulation des groupes

- Pour ajouter un groupe, l'administrateur peut modifier le fichier **/etc/group** à l'aide d'un éditeur de texte. Il peut également utiliser la commande **addgroup** ou **groupadd** (pas toujours présentes). Dan sle premier cas, il aura uniquement la ou les lignes correspondant aux groupes, à ajouter. Par exemple, la ligne :

admin : : 56 : cquoi

- Pour ajouter un utilisateur à un groupe, il suffit d'éditer le fichier **/etc/group** et de rajouter ce nom au bout de la ligne en séparant le nom des membres par une virgule.

- Pour supprimer un groupe, il suffit d'éditer le fichier **/etc/group** et d'effacer la ligne correspondante. Mais attention, n'oubliez pas de changer dans le fichier **/etc/passwd**, les numéros (GID) du groupe supprimé, si des utilisateurs y

appartenaient. Vous devez également chercher les fichiers et répertoires de ce groupe pour le changer (sinon, vos fichiers et répertoires riquent d'être inaccessibles).

8- Archives, Compression et décompression des fichiers

Les programmes sont la plupart du temps fournis compressés, c'est-à-dire sous un format plus compact permettant de réduire la taille du programme, notamment pour faciliter son téléchargement. Ce sont les outils **TAR et GZIP** qui permettent ce compactage. Grâce à l'outil **TAR**, plusieurs fichiers peuvent être simplement regroupés en une seule archive pour faciliter leur transport.

Enfin, les fichiers peuvent être regroupés puis compressés pour obtenir une archive portant l'extension **.tar.gz**.

8.1- Outil TAR : Extensions .tar et .tgz

Il est très utilisé dans le monde de Linux. Il permet de créer une archive unique contenant de nombreux fichiers et toutes leurs informations annexes (droits, propriétaires). Cette archive est plus fréquemment créée non compressée. On obtient un fichier ayant l'extension **.tar**. Puis elle peut être compressée à l'aide de **GZIP** pour obtenir un fichier avec l'extension **.tar.gz** (voir plus loin). Si on utilise l'outil **TAR** pour compresser, on obtient une archive avec **.tgz** comme extension. Voici les différentes options utilisées avec l'outil **TAR** :

-A, --catenate ou --concatenate	Ajoute des fichiers tar à une archive
-c ou --create	Crée une nouvelle archive
-d, --diff ou --compare	Cherche les différences entre une archive et les fichiers du disque
--delete	Supprime un fichier de l'archive
-f ou --file Fichier	Indique le nom du fichier archive
-r ou --append	Ajoute les fichiers à la fin d'une archive
-t ou --list	Donne le contenu de l'archive
-u ou --update	Ne met dans l'archive que les fichiers nouveaux ou plus récents que dans l'archive
-v ou --verbose	Affiche le nom des fichiers traités
-x, --extract ou --get	Extrait les fichiers d'une archive

Quelques exemples :

tar xvf archive.tar	Extrait le contenu de archive.tar dans le répertoire courant.
tar tvf archive.tar	Liste le contenu de archive.tar avec les droits, propriétaires.

tar cvf archive.tar /usr/local/program/*	Création de archive.tar dans le répertoire courant et archivage de tout le contenu de /usr/local/program (fichiers, répertoire, sous-répertoire) dans cette archive

8.2- Outils GZIP et GUNZIP Extension .gz

L'outil **GUNZIP** permet de décompresser un fichier ayant pour extension **.gz**. L'outil **GZIP**, quant à lui, permet de compresser un unique fichier. C'est pourquoi il est utilisé avec l'outil **TAR**. En effet, l'outil **TAR** regroupe différents fichiers dans une archive **.tar** et l'outil **GZIP** la compresse. L'extension **.gz** créée par **GZIP** est ajoutée à celle du fichier. On peut alors rencontrer des fichiers du type Fichier.txt.gz ou bien sûr Fichier.tar.gz. Notez bien que l'outil **GZIP** remplace le fichier à compresser par celui créé. Le fichier initial ne se trouve ainsi plus dans le répertoire. Voici les différentes options utilisées avec les outils **GZIP et GUNZIP** :

-c, --stdout ou --to-stdout	Redirige le fichier compressé vers la sortie standard et ne modifie pas l'original
-d, --decompress ou --uncompress	Décompresse l'archive (qui disparaît)
-h ou --help	Affiche les options possibles
-l ou --list	Donne des informations sur l'archive et sur les fichiers compressés
-q ou --quiet	N'affiche pas les messages d'alerte
-r ou --recursive	Suit l'arborescence des répertoires de façon récursive pour en compresser tous les fichiers (ou décompresser dans le cas de GUNZIP)
-t ou --test	Test l'intégrité de l'archive compressée

Quelques exemples :

gzip -d fichier.tar.gz ou gunzip fichier.tar.gz	Décompresse fichier.tar.gz
gzip fichier.txt	Compresse fichier.txt en le transformant en fichier.txt.gz
gzip *	Compresse tous les fichiers du répertoire courant Chaque fichier donnera un fichier gz

8.3- Extension .tar.gz

Comme vu précédemment, cette extension est obtenue par archivage à l'aide de l'outil **TAR** puis par compression à l'aide de **GZIP**. Cette extension est l'une des plus rencontrées avec l'extension RPM. Voici comment les compresser et les décompresser :

tar zxvf fichier.tar.gz ou gunzip -c fichier.tar.gz | tar xvf -	Décompresse dans le répertoire courant le fichier fichier.tar.gz et recrée les fichiers de l'archive fichier.tar
gunzip fichier.tar.gz puis tar xvf fichier.tar	Même chose
tar cvf fichier.tar /usr/local/* gzip fichier.tar	TAR crée l'archive fichier.tar, ensuite GZIP la compresse en fichier.tar.gz

9- Les droits d'accès aux ressources

Les **permissions UNIX** constituent un système simple de définition des droits d'accès aux ressources, représentées par des fichiers disponibles sur un système informatique. Elles restent le moyen le plus utilisé pour définir les droits des utilisateurs sur les systèmes de type UNIX.

Notion d'utilisateur (*user*)

Toute entité (personne physique ou programme particulier) devant interagir avec un système UNIX est authentifié sur cet ordinateur]] par un utilisateur ou *user*. Ceci permet d'identifier un acteur sur un système UNIX. Un utilisateur est reconnu par un nom unique et un numéro unique (la correspondance nom/numéro est stockée dans le fichier */etc/passwd*).

Tous les utilisateurs UNIX n'ont pas les mêmes droits d'accès à l'ordinateur (ils ne peuvent pas tous faire la même chose), et ceci simplement pour des raisons de sécurité et d'administration. Par exemple, pour éviter tout problème sur Internet, l'utilisateur qui gère le serveur HTTP n'a pas le droit d'exécuter des commandes localement, pour éviter que le serveur puisse le faire.

Certains utilisateurs ne peuvent en effet pas s'authentifier sur l'ordinateur et accéder à un interpréteur de commandes. Cela ne veut toutefois pas dire qu'ils ne peuvent rien faire sur l'ordinateur : il leur est possible de lire ou écrire des fichiers mais cela nécessite que le super utilisateur (voir plus bas) démarre un programme pour cet utilisateur. Ce mécanisme est généralement utilisé pour les démons : le super utilisateur démarre le démon et pour éviter que ce dernier puisse faire tout et n'importe quoi sur la machine, il est par exemple attribué à l'utilisateur bin.

Sur tout système UNIX, il y a un super-utilisateur, généralement appelé *root*, qui a tous les pouvoirs. Il peut accéder librement à toutes les ressources de l'ordinateur, y compris à la place d'un autre utilisateur, c'est-à-dire sous son identité. En général, du moins sur les systèmes de production, seul l'administrateur système possède le mot de passe *root*. L'utilisateur *root* porte le numéro 0.

Groupe

Un utilisateur UNIX appartient à un ou plusieurs groupes. Les groupes servent à rassembler des utilisateurs afin de leur attribuer des droits communs. Par exemple, sur un système doté d'une carte son, il y a souvent un groupe *audio* qui regroupe les utilisateurs autorisés à en faire usage.

Propriété

Tout fichier UNIX possède un propriétaire. Au départ, c'est l'utilisateur qui a créé le fichier mais il possible de le « donner » à un autre utilisateur. Seul le propriétaire du fichier et le super utilisateur (*root*) peuvent changer les droits et l'appartenance de ce fichier. Seul *root* peut s'attribuer un fichier, mais un utilisateur ordinaire peut donner un de ses fichiers à un autre utilisateur ordinaire.

Un fichier UNIX appartient aussi à un groupe. Ceci donne pleinement son sens à la notion de groupe. On définit ainsi les actions du groupe sur ce fichier. Ce groupe est souvent le groupe d'appartenance du propriétaire, mais ce n'est pas obligatoire. Tout dépend en fait de ce qu'on veut faire. On peut imaginer un scénario de délégation d'administration : le super utilisateur est propriétaire d'un fichier de configuration, mais autorise tous les utilisateurs du groupe admin (les administrateurs) à modifier ce fichier. Le fichier en question aura donc root comme propriétaire et appartiendra au groupe admin.

Rappelons que les répertoires sous UNIX sont aussi des fichiers. Les droits sur les répertoires (mais aussi les périphériques, etc.) fonctionnent exactement de la même façon que sur des fichiers ordinaires.

Droits d'accès à un fichier

À chaque fichier est associée une liste de permissions, qui déterminent ce que chaque utilisateur a le droit de faire du fichier.

Norme POSIX

Les permissions d'accès aux fichiers dans la norme POSIX sont inspirées des permissions d'accès UNIX.

Fonctionnement

Les différents droits

Les droits sur un fichier UNIX s'attribuent sur trois « actions » différentes possibles :

- la lecture (r) : on peut par exemple lire le fichier avec un logiciel
- l'écriture (w) : on peut modifier ou supprimer le fichier. Lorsque ce droit est alloué à un répertoire, il autorise la modification et la suppression des fichiers qu'il contient, quelques soient les droits d'accès des fichiers de ce répertoire (mêmes s'ils ne possèdent pas eux-mêmes le droit en écriture). Néanmoins le droit spécial *sticky bit* (voir plus bas) permet de passer outre ce comportement.
- l'exécution (x) : on peut exécuter le fichier s'il est prévu pour, c'est-à-dire si c'est un fichier exécutable. Lorsque ce droit est attribué à un répertoire, il autorise l'accès (ou ouverture) au répertoire.

On appelle parfois r, w et x des « flags » ou « drapeaux ». Sur un fichier donné, ces 3 flags doivent être définis pour son propriétaire, son groupe, mais aussi les autres utilisateurs (différents du propriétaire et n'appartenant pas au groupe).

Seuls *root* et le propriétaire d'un fichier peuvent changer ses permissions d'accès.

Représentation des droits

Cet ensemble de 3 droits sur 3 entités se représente généralement de la façon suivante : on écrit côte à côte les droits r, w puis x respectivement pour le propriétaire (u), le groupe (g) et les autres utilisateurs (o). Les codes u, g et o (u comme *user*, g comme *group* et o comme *others*) sont utilisés par les commandes UNIX qui permettent d'attribuer les droits et l'appartenance des fichiers. Lorsqu'un flag est attribué à une entité, on écrit ce flag (r, w ou x), et lorsqu'il n'est pas attribué, on écrit un '-'. Par exemple,

```
rwxr-xr--
\ /\ /\ /
 v  v  v
 |  |  droits des autres utilisateurs (o)
 |  |
 |  droits des utilisateurs appartenant au groupe (g)
 |
droits du propriétaire (u)
```

signifie que le propriétaire peut lire, écrire et exécuter le fichier, mais que les utilisateurs du groupe attribué au fichier ne peuvent que le lire et l'exécuter, et enfin que les autres utilisateurs ne peuvent que lire le fichier.

Une autre manière de représenter ces droits est sous forme binaire grâce à une clef numérique fondée sur la correspondance entre un nombre décimal et sont expression binaire :

- 0 = 000
- 1 = 001
- 2 = 010
- 3 = 011
- 4 = 100
- 5 = 101
- 6 = 110
- 7 = 111

A l'expression binaire en trois caractères sont associés les 3 types de droits (r w x) ; il suffit donc de déclarer pour chacune des catégories d'utilisateur (user, group, others) un chiffre entre 0 et 7 auquel correspond une séquence de droits d'accès. Par exemple :

- 777 donne 111 111 111 soit r w x r w x r w x
- 605 donne 110 000 101 soit r w - - - - r - x
- 644 donne 110 100 100 soit r w - r - - r - -
- 666 donne 110 110 110 soit r w - r w - r w -

Une astuce permet d'associer rapidement une valeur décimale à la séquence de droits souhaitée. Il suffit d'attribuer les valeurs suivantes pour chaque type de droit :

- lecture (r) correspond à 4
- écriture (w) correspond à 2
- exécution (x) correspond à 1

Puis on additionne ces valeurs selon qu'on veuille ou non attribuer le droit en correspondant.

Ainsi, rwx « vaut » 7 (4+2+1), r-x « vaut » 5 (4+1) et r-- « vaut » 4. Les droits complets (rwxr-xr--) sont donc équivalent à 754. Un manière directe d'attribuer les droits est de les écrire sous cette forme et d'utiliser le code à 3 chiffres résultant avec chmod (voir ci-après).

Utilisation

Pour voir quels droits sont attribués à un fichier, il suffit de taper la commande ls -l (ll) nom_du_fichier :

```
# ls –l toto
-rwxr-xr--    1 user     group     12345 Nov 15 09:19 toto
```

La sortie signifie que le fichier toto (de taille 12345) appartient à « user », qu'on lui a attribué le groupe « group », et que les droits sont rwxr-xr--. On remarque qu'il y a en fait 10 caractères sur la zone de droits. Le premier – n'est pas un droit, c'est un caractère réservé pour indiquer le type de fichier. Il peut prendre les valeurs suivantes :

- `d` : répertoire
- `l` : lien symbolique
- `c` : périphérique de type caractère
- `b` : périphérique de type bloc
- `p` : fifo
- `s` : socket
- `-` : fichier classique

Le changement de droits s'effectue avec la commande chmod ; le changement de propriétaire ou de groupe, à l'aide de la commande chown.

Changer les droits peut s'effectuer également simplement à partir du nombre à 3 chiffres calculé comme précédemment. Ainsi, pour attribuer les droits r-xr-xr-x (i.e. 555), il suffit d'exécuter :

```
chmod 555 nom_du_fichier
```

Utilisation

Les options passées à la commande chmod sont indiquées comme ceci :

`chmod` *`options modes fichiers`*

Pour un fichier : `chmod` *`[u g o a] [+ - =] [r w x] nom_du_fichier`*

Pour un répertoire (de façon récursive) :
`chmod -R` *`[u g o a] [+ - =] [r w x] nom_du_repertoire`*

Exemples

- `chmod u+rw mon_fichier` Je donne au propriétaire les droits en écriture et en lecture au fichier *mon_fichier*.
- `chmod -R a+rx mon_dossier` Je donne à tous les utilisateurs les droits en lecture et en exécution à tout ce que contient le dossier *mon_dossier*. A noter, le a est facultatif `chmod -R +rx mon_dossier` fonctionne tout aussi bien.
- `chmod 755 mon_dossier` Je donne au propriétaire tous les droits, aux membres du groupe et aux autres les droits de lecture et d'accès. C'est un droit utilisé traditionnellement sur les répertoires.
- `chmod 644 mon_fichier` Je donne au propriétaire les droits de modification et lecture, aux membres du groupe et aux autres uniquement les droits de lecture. C'est un droit utilisé traditionnellement sur les fichiers.

Options

`chmod` a un certain nombre d'options qui peuvent modifier le résultat. Certaines de ces options sont :

- `-R`: récursivité, change les modes de tous les fichiers dans les sous-répertoires.
- `-v`: verbeux. Affiche la liste de tous les fichiers en cours de modification.

Modes

Pour chaque fichier donné, les permissions s'appliquent au propriétaire du fichier (u), aux utilisateurs dans le groupe du fichier (g) ou à tous les autres utilisateurs (o). Pour appliquer les modifications à tous en une seule fois, on utilise la commande (a) pour all.

Les modes peuvent être spécifiés de deux façons, avec des lettres ou avec des nombres en octal. Pour les lettres, il existe des opérateurs de changement comme `+`, un mode d'ajout, `=` définit le mode et `-`, enlever le droit du mode. Pour l'octal il faut additionner les nombres pour chaque type de possesseur.

Les permissions sont (valeurs octales entre parenthèses) :

- `r` (4) : autorisation de lecture
- `w` (2) : autorisation d'écriture
- `x` (1) : autorisation d'exécution. Il faut noter, que la permission d'exécution régit également l'accès à un répertoire (si l'exécution n'est pas autorisée sur un répertoire, on ne peut faire un chdir (commande cd) sur ce répertoire).

10- Redirections et Pipes

Toute commande possède un flux standard :
- une entrée (ou Input) standard : par défaut le clavier
- une sortie (ou Output) standard : par défaut l'écran
- une erreur standard : par défaut l'écran (comme la sortie standard).

Seulement ce flux standard n'est pas forcément ce qu'il y a de plus pratique, on peut vouloir une sortie autre qu'à l'écran, ou entrer les données autrement qu'au clavier (ce qui peut être

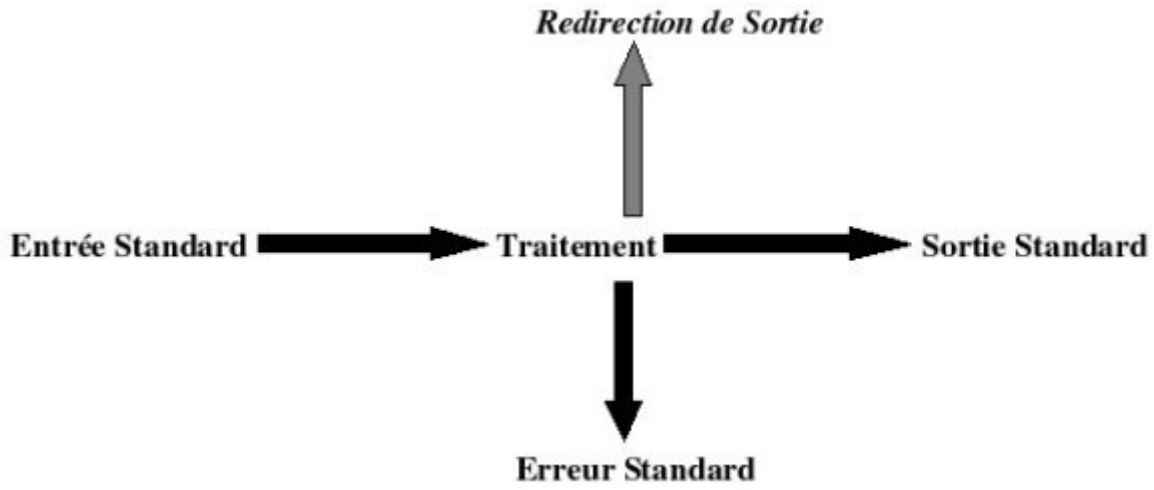
fastidieux). Pareil pour les messages d'erreur que l'on peut vouloir obtenir sous forme de fichier log par exemple. Pour pouvoir dévier le flux standard on utilise des redirections.

L'exemple le plus simple est la redirection de sortie. Si par exemple on veut faire, un ls du dossier **/usr/bin**, le nombre de fichiers est tellement long qu'ils dépassent de l'écran. Si on veut pouvoir regarder tranquillement la liste des fichiers sous forme de fichier texte, il faut rediriger la sortie standard de l'écran vers un fichier. Pour cela on utilise le signe > :

ls /usr/bin > listefichiers (sous linux un fichier n'a pas besoin d'avoir une extension, on n'en met que par commodité pour l'interopérabilité et pour savoir quelle application va pouvoir ouvrir le fichier).

Le résultat de cette commande est que rien n'apparaît à l'écran (la sortie standard n'est plus utilisée), sauf les messages d'erreurs (et il y en aura sûrement si vous lancez cette commande en tant que simple utilisateur, j'expliquerai pourquoi ultérieurement) et si vous allez regarder ensuite dans le dossier où vous vous trouvez (souvent votre home ou dossier personnel) vous y verrez le fichier **listefichiers**. Ouvrez le avec un éditeur de texte et vous y verrez le résultat de la commande ls.

Voici sous forme de schéma le flux standard et la redirection de sortie :



On peut rediriger vers un fichier, mais aussi vers un terminal, ou un périphérique (imprimante par exemple). Pour rediriger le résultat de la commande echo bonjour vers le premier terminal virtuel (**tty1**) il faut taper :
echo bonjour > /dev/tty1 (les terminaux virtuels vont le plus souvent de **tty1** à **tty6**)

Vous ne verrez rien sur la console d'où vous avez lancer la commande echo, par contre si vous faîtes **Ctrl+Alt+F1** vous verrez affiché **bonjour**.

De même que l'on peut rediriger la sortie, on peut rediriger l'entrée. Par exemple si on veut compter le nombre de lignes, de colonnes et de mots d'un texte que l'on a déjà tapé et mis dans un fichier texte, il suffit d'indiquer à la commande wc une autre entrée que l'entrée standard. Pour rediriger l'entrée d'une commande on utilise le signe < , ce qui nous donnerait par exemple :
wc < fichiertexte

Et si on voulait que le résultat de la commande wc sur ce fichier texte sorte lui aussi sous forme de fichier ? Rien de plus simple il suffit de rediriger l'entrée ET la sortie :

```
wc < fichiertexte > resultat
```

Avec cette ligne de commande, wc ira chercher son entrée dans le fichier **fichiertexte** et sortira son résultat dans le fichier **resultat**. Vous n'aurez donc pas à entrer le texte au clavier et rien ne s'affichera à l'écran.

Maintenant supposons que vous lancez un programme en mode console, et celui-ci plante à chaque fois en vous sortant un monceau de messages d'erreurs à la queue leu leu, et la liste des messages d'erreurs est tellement longue qu'elle dépasse de l'écran. Cela n'est guère pratique et pour pouvoir analyser les messages d'erreurs afin de corriger le problème il faut pouvoir lire tranquillement ceux-ci. Pour cela rien de plus simple, nous allons rediriger le flux d'erreur. Pour cela on utilise **2>** , ce qui donne par exemple avec le programme **k3b** :

```
k3b 2> k3b-erreur
```

Aucun message d'erreur n'apparaîtra à l'écran, par contre si vous ouvrez avec un éditeur de texte le fichier **k3b-erreur**, vous y verrez tous les messages d'erreur, que vous pourrez lire et analyser tranquillement. Pourquoi **2>** comme signe de redirection ? Tout simplement pour le distinguer du signe > de redirection de la sortie standard. Nous verrons plus loin à quoi correspond le **2** devant le >. Comme pour la sortie standard on peut rediriger en même temps l'entrée de la commande. On peut même rediriger les trois à la fois :

```
wc < fichiertexte > nbmots 2> wc-erreur
```

La commande wc va aller chercher son entrée dans le fichier **fichiertexte**, va sortir son résultat dans le fichier **nbmots**, et s'il y a des erreurs elles seront écrites dans le fichier **wc-erreur**. Rien n'apparaîtra à l'écran, et il n'y aura pas besoin d'entrée au clavier.

Il existe aussi un signe de sortie permettant la concaténation, il s'agit du signe >>. Par exemple si on veut ajouter le contenu d'un fichier à un autre, on peut utiliser la commande `cat` avec ce signe de concaténation, par exemple :

```
cat fichier1 >> fichier2
```

La sortie de la commande `cat` (c'est-à-dire le contenu du **fichier1**) est redirigée vers le **fichier2** mais sans l'écraser (ce que ferait une redirection de sortie classique). Le contenu de **fichier1** est ajouté après celui de **fichier2**, et le tout est enregistré sous le nom de **fichier2**.

Nous avons vu comment rediriger l'entrée, la sortie et l'erreur standard d'une commande, et comment concaténer plusieurs fichiers texte, ce qui est très pratique. Maintenant si on veut que la sortie d'une commande devienne l'entrée d'une autre commande ? On peut passer par un fichier intermédiaire, par exemple :

```
ls –l /usr/bin > listefichiers
wc –l < listefichiers (n'affiche que le nombre de lignes)
```

On aura ainsi le nombre de dossiers et fichiers contenus dans **/usr/bin** (puisqu'avec l'option –l la commande `ls` affiche un fichier ou dossier par ligne, et que `wc` ne va compter que les lignes). Mais cela crée un fichier qu'il faudra détruire après lorsqu'il ne servira plus à rien, et ce n'est pas très pratique ni très performant.

Pour économiser la création, puis la destruction de ce fichier temporaire, et faire tenir les deux commandes en une seule ligne, nous allons utiliser un tube (ou *pipe* en anglais, terme le plus souvent employé), celui-ci est symbolisé par le signe | (obtenu le plus souvent par un **Ctrl+Alt+ <touche1>** du clavier (au dessus des lettres) et souvent représenté sur les claviers par deux traits verticaux alignés au lieu d'un seul grand trait vertical) ce qui nous donnerait dans notre exemple :

ls –l /usr/bin | wc –l

Le pipe permet de simplifier la ligne de commande obtenue et optimise celle-ci. Ici pas de création de fichier sur le disque dur, pas d'occupation d'espace disque, tout se passe en mémoire : le flux de sortie de la première commande est envoyé comme entrée de la seconde, puis est automatiquement détruit dès qu'il n'est plus nécessaire.

Le pipe n'empêche pas la redirection, on peut très bien combiner les deux par exemple :
ls –l /usr/bin | wc –l > nbfichiers 2> fichier-erreur

Ainsi le résultat de la ligne de commande n'apparaîtra pas à l'écran, et sera redirigé vers le fichier **nbfichiers**, et les messages d'erreur seront redirigés vers le fichier **fichier-erreur**.

Si on veut néanmoins le résultat de la commande ls dans un fichier en plus du résultat de la commande wc, alors il va falloir dédoubler le flux de sortie de ls, comme le ferait un tuyau en forme de **T** en plomberie. Pour dédoubler le flux on utilise la commande tee :
ls –l /usr/bin | tee listefichiers | wc –l

On aura ainsi à l'écran le résultat de la commande wc appliquée sur le résultat de la commande ls, ET la création du fichier **listefichiers** contenant le résultat de la commande ls.

Si on veut que la commande tee n'écrase pas ce qui se trouve dans le fichier de destination, mais ajoute des données (comme avec le signe de concaténation >> et la commande cat), il faut utiliser l'option –a. Cela peut servir par exemple à contrôler qui se connecte sur un poste de travail au fur et à mesure de la journée, tout en affichant régulièrement le nombre de connectés. Pour cela on va utiliser la commande who, et la commande wc, avec deux pipes et la commande tee :
who | tee –a qui | wc –l

Il suffit de lancer cette ligne de commande régulièrement dans la journée (grâce à un planificateur comme cron dont je parlerai dans le Cours d'Administration), et au fur et à mesure de la journée le résultat de la commande who sera ajouté au fichier **qui**, et la commande wc –l affichera le nombre d'utilisateurs connectés (puisque who sort une ligne par utilisateur connecté).

11- Gestion des processus : commande ps

1. La commande "**ps**" :
2. La commande **ps** permet de connaître les processus actifs à un moment donné :
3. **[delcros@mistra delcros]$ ps**
4. PID TTY STAT TIME COMMAND
5. 341 p1 S 0 : 00 bash
6. 344 p2 S 0 : 00 bash
7. 1039 p3 S 0 : 00 bash
8. 1219 p3 R 0 : 00 ps

Le "PID" est l'identificateur d'un processus, c'est un nombre. Chaque processus est identifié dans le système par un nombre unique.

Le "TTY" indique à quel port de terminal est associé le processus.

"STAT" indique l'état dans lequel se trouve le processus. Dans l'exemple, trois processus sont endormis (S comme "sleep"), et un processus en cours d'exécution (R comme "run"). Le processus qui est en cours d'exécution n'est autre que la commande "**ps**" que nous venons de lancer.

Le "TIME" indique depuis combien de temps le processus utilise les ressources du microprocesseur.

Le "COMMAND" précise, comme son nom l'indique, la commande dont l'état est décrit par PID, TTY, STAT et TIME.

Ceci dit, une simple commande "**ps**" n'indique pas tous les processus du système. Le simple fait de lancer **ps** nous a juste indiquer les processus associés à un terminal et qui dépendent de l'utilisateur courant (ici "delcros").

En fait, il est tout a fait probable que d'autres processus non liés à un terminal aient été lancés par "delcros". J'en suis d'ailleurs sur, puisque actuellement j'utilise emacs pour réaliser cette modeste page de documentation et que pour visualiser le résultat, j'utilise netscape :
[delcros@mistra delcros]$ ps -x

```
  PID TTY STAT  TIME COMMAND
  240  ?  S    0:01 /usr/X11R6/bin/fvwm2
  246  ?  S    0:00 /usr/X11/bin/xautolock -corners ++++ -time 5 -locker /usr/X
  247  ?  S    0:00 /usr/X11/bin/unclutter -idle 3
  253  ?  S    0:00 /usr/local/bin/Periodic
  254  ?  S    7:34 emacs --background grey79 -geometry 80x58+-4+-11
  257  p0 S    0:00 bash
  258  p2 S    0:00 bash
  259  p1 S    0:00 bash
  272  ?  S    0:00 /usr/lib/emacs/19.34/i386-gnu-linux/emacsserver
 2134  ?  S    0:00 /usr/bin/ispell -a -m -d francais
 6431  p0 S    1:03 /usr/lib/netscape/netscape-navigator
 6441  p0 S    0:00 (dns helper)
 6741  p0 R    0:00 ps -x
```

Les commandes qui ne sont pas associées à un terminal sont reconnaissables par le point d'interrogation qui rempli le champs TTY.

1. Si vous voulez connaître tous les processus de la machine de tous les utilisateurs, il suffit d'utiliser l'option **ax**. Si en plus vous voulez connaître les utilisateurs associés à chaque processus, il vous suffit d'utiliser l'option **aux**. Vous verrez alors plusieurs colonnes s'ajouter dont "USER" qui indique à quel utilisateur appartient le processus. "%CPU" indique en pourcentage les ressources du microprocesseur utilisées par le processus. "%MEM" montre en pourcentage les ressources en mémoire vive utilisées par le processus. "RSS" donne réellement la mémoire utilisée en kilobytes par le La commande "**kill**" :

2. La commande "**kill**" permet d'expédier un signal à un processus en cours.
3. Sa syntaxe est la suivante :
4. **kill [options] PID**

Par exemple, si j'ai lancé une connexion à l'Internet en PPP, un processus pppd sera en cours. Pour tuer le processus, je peux d'abord faire un **ps -ax** pour connaître le numero du PID de pppd et ensuite si par exemple le PID est 592, je peux tuer la connexion en faisant :
[root@mistra delcros]# kill 592

Vous remarquerez que je suis logué en utilisateur "root" pour faire ceci, en effet le processus pppd appartenait à l'utilisateur "root" et un autre utilisateur ne peut pas lui expédier de signal.

Si un processus vous résiste, c'est à dire que vous n'arrivez pas à le tuer, vous devez utiliser la commande : **kill -9 PID** (PID étant toujours le numéro de de processus).

La commande "**killall**" permet aussi de tuer un processus mais au lieu d'indiquer le PID vous indiquerez le nom du processus.

Mais **attention**, plusieurs processus peuvent utiliser la même commande. Ainsi, si vous tapez :

[delcros@mistra delcros]# killall grep

Vous tuerez tous les processus qui contiennent la commande grep. Je vous recommande donc d'utiliser l'option "-i" qui vous demande une confirmation avant de tenter d'arrêter un processus

processus. "START" indique l'heure à laquelle le processus a été lancé.

12- Editeur de texte vi

12.1 Lancer l'éditeur VI

L'éditeur VI permet à l'utilisateur de créer de nouveaux fichiers ou d'éditer des fichiers existants. La commande pour lancer l'éditeur VI est vi, suivie par le nom de fichier. Par exemple, pour éditer le fichier nommé temporaire, vous devez tapez vi temporaire et ensuite retour. Vous pouvez lancer VI sans nom de fichier, mais quand vous voudrez sauvez votre travail, vous devrez indiquer plus tard à VI sous quel nom il devra le sauver.

Quand vous lancez VI pour la première fois, vous voyez un écran rempli de tildes (un tilde ressemble à ceci : ~) sur le côté gauche de l'écran. Les lignes vides au-delà de la fin du fichier sont montrées de cette manière. En bas de votre écran, si vous avez spécifié un nom de fichier existant, le nom et la taille de ce fichier sont affichés, comme ceci :

"nom_de_fichier" 21 lines, 385 characters
Si le fichier que vous avez spécifié n'existe pas, il vous est indiqué qu'il s'agit d'un nouveau fichier, comme ceci :
"newfile" [New file]

Si vous lancez VI sans nom de fichier, la dernière ligne de l'écran restera vide. Si l'affichage ne correspond à ces indications, votre terminal est sans doute mal réglé. Tapez :q et "retour" pour sortir de VI, et régler votre type de terminal.

12.2 Sortir de VI

Maintenant que vous savez entrer dans VI, ce serait une bonne idée de savoir en sortir. L'éditeur VI a deux modes et pour sortir de VI, vous devez être en mode commande. Appuyer sur la touche appelée "Escape" ou "Esc" (Si votre terminal n'a pas une telle touche, essayez ^[, ou contrôle-[) pour passer en mode commande. Si vous êtes déjà en mode commande quand vous tapez "Escape", cela peut "biper", ne vous inquiétez pas, vous êtes encore dans le mode commande.

La commande pour quitter VI est :q. Une fois en mode commande, tapez deux points, et 'q', suivi de retour. Si votre fichier a été modifié d'une façon ou d'une autre, l'éditeur vous préviendra, et ne vous laissera pas quitter. Pour éviter ce message, la commande est :q!. Cela vous laisse quitter VI sans sauvegarder les changements.

Bien sûr, normalement dans un éditeur, vous devriez vouloir sauvegarder les changements que vous avez faits. La commande pour enregistrer le contenu de l'éditeur est :w. Vous pouvez combiner cette commande avec la commande pour quitter, soit :wq. Vous pouvez spécifier un nom de fichier différent pour sauver sous ce nom après le :w. Par exemple, si vous voulez sauvegarder le fichier sur lequel vous travaillez sous le nom nom_de_fichier_2, vous devez taper :w nom_de_fichier_2 et retour.

Une autre façon de sauver vos changements et quitter VI est la commande ZZ. Quand vous êtes en mode commande, taper ZZ est équivalent à :wq. Si des changements ont été faits dans le fichier, ils seront sauvés. C'est la façon la plus simple de quitter l'éditeur, avec seulement deux appuis de touches.

12.3 Travailler avec les deux modes de VI

La première chose que beaucoup d'utilisateurs apprennent à propos de l'éditeur VI est qu'il a deux modes: commande et insertion. Le mode commande permet l'entrée de commandes pour manipuler du texte. Ces commandes sont en général longues d'un ou deux caractères, et peuvent être entrées en peu de frappe de touche. Le mode insertion met tout ce qui est tapé sur le clavier dans le fichier courant.

VI démarre en mode commande. Plusieurs commandes mettent l'éditeur VI en mode insertion. Les commandes les plus couramment utilisées sont a et i. Ces deux commandes sont décrites plus bas. Une fois en mode insertion, vous en sortez en tapant la touche escape. Si votre terminal n'a pas de touche escape, ^[devrait convenir (contrôle-[). Taper escape pendant que vous êtes déjà en mode commande ne fait pas sortir l'éditeur du mode commande. Un bip peut vous indiquer que vous êtes déjà dans ce mode.

12.4 Taper des commandes en mode commande

Les commandes du mode commande sont généralement au format suivant : (Les arguments optionnels sont donnés entre crochet.

[quantité] commande [lieu]

La plupart des commandes font un caractère de long, en particulier celles qui utilisent le caractère de contrôle. Les commandes décrites dans cette section sont celles que l'on utilise le plus avec l'éditeur VI.

La quantité est définie par n'importe quel caractère de 1 à 9. Par exemple, la commande x efface un caractère sous le curseur. Si vous tapez 23x en mode commande, cela va effacer 23 caractères.

Certaines commandes utilisent un paramètre optionnel lieu, où vous pouvez spécifier le nombre de lignes ou la partie du document concernée par la commande. Le paramètre lieu peut aussi être n'importe quelle commande qui déplace le curseur.

12.5 Quelques commandes VI simples

Ceci est un simple groupe de commande pour permettre à un utilisateur débutant avec VI de commencer. D'autres commandes utiles, seront présentées ultérieurement.

a
: fait entrer en mode *insertion*, les caractères tapés ensuite seront insérés après le curseur. Si vous spécifiez un nombre, tout le texte qui a été inséré sera répété ce nombre de fois

h
: déplace le curseur sur le caractère précédent.

i
: entre en mode *insertion*, Le caractère tapé ensuite sera inséré avant la position actuelle du curseur. Si vous spécifiez un nombre, tout le texte qui a été inséré sera répété ce nombre de fois

j
: déplace le curseur d'une ligne vers le bas.

k
: déplace le curseur d'une ligne vers le haut.

l
: déplace le curseur d'un caractère vers la droite.

r
: remplace le caractère sous le curseur. Un nombre spécifie le nombre de caractères à remplacer.

u
: annule le dernier changement dans le fichier. Taper de nouveau u rétablit le changement.

x
: efface le caractère sous le curseur. *Nombre* spécifie combien de caractères il faut effacer. Les caractères seront effacés après le curseur.

12.6 Les "buffers" texte de VI

L'éditeur VI a 36 "buffers" pour conserver des morceaux de texte, ainsi qu'un "buffer" principal. Chaque fois, qu'un bout de texte est effacé ou copié depuis le fichier, il est placé dans le "buffer" principal. Beaucoup d'utilisateurs de VI utilisent rarement les autres buffers,

et se débrouillent sans eux. Le bloc de texte peut aussi être placé dans un autre buffer demandé en utilisant la commande ". Après avoir tapé ", une lettre ou un chiffre identifiant le buffer doit être entré. Par exemple, la commande : "mdd remplit le buffer **m** et efface la ligne courante. De même, du texte peut être collé avec les commandes p ou P. "mp colle le contenu du buffer **m** après la position actuelle du curseur. Pour toutes les commandes utilisées dans les deux sections suivantes, ces buffers peuvent être spécifiés pour un stockage temporaire de mots ou de paragraphes.

Couper et Copier

La commande la plus couramment utilisée pour couper est d. Cette commande efface du texte dans le fichier. La commande est précédée par un *nombre* optionnel et suivie par une indication de déplacement. Si vous doublez la commande en tapant dd, la ligne courante est effacée. Voici quelques combinaisons :

d^
: efface de la position actuelle du curseur jusqu'au début de la ligne.

d$
: efface de la position actuelle du curseur jusqu'à la fin de la ligne.

dw
: efface de la position actuelle du curseur jusqu'à la fin du mot

3dd
: efface 3 lignes à partir de la position actuelle du curseur vers le bas.

La commande y (commande de copie) opère comme la commande d en prenant du texte du fichier sans effacer le texte.

Coller

Les commandes pour coller sont p et P. Elles diffèrent seulement par la position où elles collent le texte par rapport au curseur. p colle le contenu du buffer spécifié ou du buffer général après le curseur, tandis que P le colle avant le curseur. *nombre* avant la commande, colle le texte le nombre de fois demandé.

12.6 Recherche de mots ou de caractères

L'éditeur VI peut faire deux types de recherche : chaînes de caractères ou caractères. Pour une recherche de chaîne, les commandes / et ?sont utilisées. Quand vous lancez ces commandes, la commande que vous venez de taper sera affichée sur la ligne du bas, où vous taperez la chaîne que vous cherchez. Ces deux commandes diffèrent uniquement par la direction dans laquelle la recherche est entreprise. La commande / cherche vers la fin du fichier, tandis que la commande ? cherche vers le début du fichier (de bas en haut). Les commandes n et N répètent la précédente recherche respectivement dans le même sens et dans le sens opposé. Certains caractères ont une signification particulière pour VI, et doivent donc être précédés d'un antislash (\) pour faire partie de l'expression recherchée.

Caractères spéciaux :

^
: Début de ligne. (Au début de l'expression recherchée.)

.
: Correspond à un caractère simple.

*
: Correspond à zéro ou plusieurs fois le caractère précédent.

$
: Fin de ligne (A la fin de l'expression recherchée.)

[
: Correspond au début d'un groupe d'expressions correspondantes ou non. Par exemple /f[iae]t correspond à chacun de ces mots : fit fat fet. Dans ce cas, il correspond uniquement à ceux-là. /a[^bcd] ne correspond à aucun d'eux mais à tout ce qui contient "a" et une lettre autre que "b","c","d".

<
: Peut être inséré dans une expression précédée d'un antislash pour rechercher le début ou la fin d'un mot. Par exemple /\<le\> va chercher seulement le mot "le" mais pas les mots comme "leur" ou "fleur".

>
: Voir la description du caractère '<' au dessus.

La recherche de caractères cherche une ligne à l'intérieur de laquelle se trouve le caractère entré après la commande. Les commandes `f` et `F` cherchent un caractère uniquement sur la ligne courante. `f` cherche vers l'avant et `F` cherche vers l'arrière et le curseur se déplace sur la position du caractère trouvé.

Les commandes `t` et `T` cherchent un caractère seulement sur la ligne courante, mais avec `t`, le curseur se déplace sur la position avant le caractère, et `T` cherche depuis la fin de la ligne vers la position après le caractère .

Ces deux commandes peuvent être répétées en utilisant les commandes `;` ou, (`;` répète la dernière recherche de caractère dans la même direction, alors que `,` la répète en sens inverse.)

Si l'option "ic" (ou ignorecase) est activée (tapez `:set ic`) la recherche est insensible à la casse.

12.7 Résumé des commandes VI

Couper et coller/effacer du texte

"
: Spécifie le buffer à utiliser avec les commandes se servant d'un buffer. Faites suivre le " par la lettre ou le nombre, qui correspond au buffer.

D
: Efface (Delete) de la position actuelle du curseur à la fin de la ligne.

P
: Colle (Paste) le buffer indiqué avant la position actuelle du curseur. Si aucun buffer n'est indiqué (avec la commande ") alors 'P' utilise le buffer général.

X
: Efface le caractère avant le curseur.

Y
: Copie (Yank) la ligne courante dans le buffer indiqué. Si aucun buffer n'est indiqué, le buffer général est utilisé.

d
: Efface (Delete) jusqu'à *lieu*. "dd" efface la ligne courante. Un *nombre* efface ce nombre de lignes. Tout ce qui est effacé est placé dans le buffer spécifié avec la commande ". Si aucun buffer n'est spécifié, le buffer général est utilisé.

p
: Colle (Paste) le buffer spécifié après la position actuelle du curseur ou la ligne. Si aucun buffer n'est spécifié (avec la commande ") 'p' utilise le buffer général.

x
: Efface le caractère sous le curseur. Un *nombre* indique combien de caractères doivent être effacés. Les caractères seront effacés après le curseur.

y
: Copie (Yank) jusqu'à *lieu*, en mettant le résultat dans un buffer. "yy" copie la ligne courante. Un *nombre* copie ce nombre de lignes. Le buffer peut être spécifié avec la commande ".Si aucun buffer n'est spécifié, le buffer général est utilisé.

Insérer du texte

A
: Joint (Append) à la fin de la ligne courante.

I
: Insère à partir du début de la ligne.

O
: (la lettre)Entre en mode *insertion* sur une nouvelle ligne au-dessus de la position courante du curseur.

a
: Entre en mode *insertion*, les caractères tapés seront ajoutés après la position courante du curseur. Un *nombre* insère tout le texte ce nombre de fois.

i
: Entre en mode *insertion*, les caractères tapés seront insérés avant la position courante du curseur. Un *nombre* insère tout le texte ce nombre de fois.

o
: Entre en mode *insertion* sur une nouvelle ligne en dessous de la position courante du curseur.

Déplacer le curseur dans le fichier

^B
: Monte d'une page. Un *nombre* fait monter de ce nombre de pages.

^D
: Descend d'une demi fenêtre. Un *nombre* fait descendre de ce nombre de lignes.

^F
: Descend d'une page. Un *nombre* fait descendre de ce nombre de pages.

^H
: Déplace le curseur d'un espace vers la gauche. Un *nombre* fait déplacer de ce nombre d'espaces.

^J
: Descend le curseur d'une ligne dans le même colonne. Un *nombre* fait descendre de ce nombre de lignes.

^M
: Déplace vers le premier caractère de la ligne suivante.

^N
: Descend le curseur d'une ligne dans le même colonne. Un *nombre* fait descendre de ce nombre de lignes.

^P
: Monte le curseur d'une ligne dans le même colonne. Un *nombre* fait monter de ce nombre de lignes.

^U
: Monte d'une demi fenêtre. Un *nombre* fait monter de ce nombre de lignes.

$
: Déplace le curseur à la fin de la ligne courante. Un *nombre* le fait se déplacer à la fin des lignes suivantes.

%
: Déplace le curseur sur la parenthèse ou accolade correspondante.

^
: Déplace le curseur sur le premier caractère non-blanc.

(
: Déplace le curseur au début de la phrase.

)
: Déplace le curseur au début de la phrase suivante.

{
: Déplace le curseur au paragraphe précédent.

}
: Déplace le curseur au paragraphe suivant.

|
: Déplace le curseur à la colonne indiquée par le *nombre*.

+
: Déplace le curseur sur le prochain caractère non-blanc de la ligne suivante.

-
: Déplace le curseur sur le prochain caractère non-blanc de la ligne courante.

0
: (Zéro) Déplace le curseur sur la première lettre de la ligne courante.

B
: Déplace le curseur d'un mot en arrière, en sautant la ponctuation.

E
: Avance à la fin du mot, en sautant la ponctuation.

G
: Aller (Go) à la ligne indiquée par le *nombre*. Si aucun *nombre* n'est donné, il va à la fin du fichier.

H
: Déplace le curseur sur le premier caractère non-blanc en haut de l'écran.

L
: Déplace le curseur sur le premier caractère non-blanc en bas de l'écran.

M
: Déplace le curseur sur le premier caractère non-blanc au milieu de l'écran.

W
: Avance au début d'un mot,en sautant la ponctuation.

b
: Recule le curseur d'un mot. Si le curseur est au milieu d'un mot, place le curseur sur le premier caractère du mot.

e

Avance le curseur d'un mot. Si le curseur est au milieu d'un mot, place le curseur sur le dernier caractère du mot.

h
Déplace le curseur sur le caractère de gauche.

j
Descend le curseur d'une ligne.

k
Monte le curseur d'une ligne.

l
Déplace le curseur sur le caractère de droite.

w
Déplace le curseur sur le mot suivant. Si le curseur est au milieu d'un mot, place le curseur sur le premier caractère du mot suivant.

Déplacer le curseur dans l'écran

^E
Déplace l'écran d'une ligne vers le haut. Un *nombre* fait monter de ce nombre de lignes.

^Y
Déplace l'écran d'une ligne vers le bas. Un *nombre* fait descendre de ce nombre de lignes.

z
Redessine l'écran avec les options suivantes. "z" place la ligne courante en haut de l'écran; "z." place la ligne courante au centre de l'écran; et "z-" place la ligne courante en bas de l'écran. Si vous spécifiez un *nombre* avant la commande 'z', il agit sur la ligne spécifiée. Par exemple, "14z." place la ligne 14 au centre de l'écran.

Remplacer du texte

C
Change de la position du curseur jusqu'à la fin de la ligne.

R
Remplace les caractères de l'écran par les caractères entrés, s'arrête en tapant la touche Escape.

S
Change une ligne entière.

c
Change jusqu'à <where>. "cc" change la ligne courante. Un *nombre* fait changer ce nombre de lignes.

r
Remplace un caractère sous le curseur. Indiquer un *nombre* fait remplacer ce nombre de caractères.

s
Substitue un caractère sous le curseur et passe en mode insertion. Spécifier un *nombre* pour substituer ce nombre de caractères. Un signe dollar($) sera ajouté au dernier caractère substitué.

Rechercher du texte ou des caractères

,
: Répète la dernière commande f, F, t ou T dans la direction opposée.

/
: Recherche vers le bas dans le fichier la chaîne spécifiée après le /.

;
: Répète la dernière commande f, F, t ou T.

?
: Recherche vers le haut dans le fichier la chaîne spécifiée.

F
: Recherche vers l'avant dans la ligne courante le caractère spécifié après la commande 'F'. S'il l'a trouvé, déplace le curseur sur sa position.

N
: Répète la dernière recherche faite par '/' ou '?', mais dans la direction opposée.

T
: Recherche vers l'arrière dans la ligne courante le caractère spécifié après la commande 'T', et se place sur la colonne suivante s'il l'a trouvé.

f
: Recherche dans la ligne courante le caractère spécifié après la commande 'f'. S'il l'a trouvé, déplace le curseur sur sa position.

n
: Répète la dernière recherche faite par '/' ou '?'.

t
: Cherche dans la ligne courante le caractère spécifié après la commande 't' et se place sur la colonne précédant le caractère, s'il l'a trouvé.

Manipulation des caractères/Formatage de ligne

~
: Inverse la casse du caractère sous le curseur.

<
: Décale jusqu'à *lieu* d'une tabulation vers la gauche. "<<" Décale la ligne courante et peut être répété avec un *nombre*.

>
: Décale jusqu'à *lieu* d'une tabulation vers la droite. ">>" décale la ligne courante et peut être répété avec un *nombre*.

J
: Joint la ligne courante à la suivante. Un *nombre* fait joindre ce nombre de lignes.

Sauvegarder et quitter

^\
: Sort du mode "VI" et passe en mode "EX". L'éditeur EX est l'éditeur en ligne de commande sur lequel VI est construit. La commande EX pour retourner dans VI est ":vi".

Q
: Sort du mode "VI" et passe en mode "EX". L'éditeur EX est l'éditeur en ligne de commande sur lequel VI est construit. La commande EX pour retourner dans VI est ":vi".

ZZ
: Quitte l'éditeur, en sauvant tout les changements effectués.

Divers

^G
: Affiche le nom du fichier courant et son statut.

^L
: Vide et redessine l'écran.

^R
: Redessine l'écran en retirant les mauvaises lignes.

^[
: Touche d'échappement. Annule les commandes partiellement tapées.

^^
: Retourne au dernier fichier édité.

!
: Exécute un shell. Si un *lieu* est spécifié, le programme qui est exécuté avec ! utilise les ligne(s) spécifiée(s) comme entrée standard, et va les remplacer par la sortie standard du programme exécuté. "!!" exécute un programme utilisant la ligne courante comme entrée. Par exemple "!4jsort" va prendre 5 lignes à partir de la position courante du curseur et exécuter sort. Après avoir tapé la commande, il y aura un point d'exclamation où vous pourrez taper la commande.

&
: Répète la précédente commande ":s".

.
: Répète la dernière commande qui a modifié le fichier.

:
: Commence une commande de l'éditeur EX. La commande est exécutée une fois que l'utilisateur a tapé retour.(voir section ci-dessous.)

@
: Tape la commande stockée dans le buffer spécifié.

U
: Restaure la ligne courante dans l'état où elle se trouvait avant que le curseur aille à la ligne.

m
: Marque la position courante du curseur avec le caractère spécifié après la commande 'm'.

u
: Annule (Undo) le dernier changement dans le fichier. Retaper 'u' va refaire le changement.(NDT : avec Vim,'u' annule le changement précédent et '.' le rétablit.).

Les commandes EX

L'éditeur VI est basé sur un autre éditeur, nommé EX. L'éditeur EX édite seulement par ligne. Depuis l'éditeur VI, vous utilisez la commande : pour entrer une commande EX. La liste donnée ici n'est pas complète, mais les commandes proposées sont les plus utilisées. Si plus d'une ligne doit être modifiée par certaines commandes (comme ":s" et ":w"), un intervalle doit être spécifié avant la commande. Par exemple, pour substituer de la ligne 3 à la ligne 15, la commande est ":3,15s/from/this/g".

:ab chaîne chaînes
: Abréviation. Si un mot correspondant à chaîne est tapé dans VI, l'éditeur insère automatiquement les mots correspondants. Par exemple, abréviation ":ab usa United

States of America" va insérer les mots, "United States of America" partout où le mot "usa" est tapé.

:map touches new_seq
Mapping. Ceci vous permet de remplacer une touche ou une séquence de touches par une autre touche ou séquence de touches.

:q
Quitte VI. Si des changements ont été faits, l'éditeur va afficher un message d'avertissement.

:q!
Quitte VI sans sauvegarder les changements.

:s/*pattern*/*to_pattern*/*options*
Substitution. Ceci substitue le modèle spécifié par la chaîne dans to_pattern. Sans option, il substitue seulement la première occurrence du modèle. Si un 'g' est spécifié, toutes les occurrences seront substituées. Par exemple, la commande ":1,$s/Dwayne/Dwight/g" remplace toutes les occurrences de "Dwayne" par "Dwight".

:set [all]
Définit les options de VI et EX. La commande ":set all" affiche les options possibles. (Voir la section sur la personnalisation de VI pour quelques options.)

:una string
Retire l'abréviation définie précédemment par ":ab".

:unm keys
Retire le mapping défini par ":map".

:vi
Commence l'édition d'un nouveau fichier. Si les changements n'ont pas étés sauvegardés, l'éditeur vous le signale.

:w
Sauvegarde (Write out) le fichier courant.

:w nom_de_fichier
Ecrit (Write) le buffer dans le fichier spécifié.

:w >> filename
Ajoute le contenu du buffer au fichier.

:wq
Enregistre le buffer et quitte.

Glossaire

a2p : Convertisseur de awk vers Perl
a2aps : Conversion d'ASCII en PostScript
adb : Debugueur
adduser : Ajouter un nouvel utilisateur

alias : Définition d'abréviation de commandes
apropos : Retourne de l'aide sur un mot clef (man -k)
ar : Archivage de fichiers (biblioth\xe9ques .a)
arp : Affiche la table de conversion des adresses Internet
at : exécution d'une commande a un moment précis
atq : Affiche la liste des commandes en attente pour un utilisateur
atrm : Efface la totalité des commandes en attente
as : Compilateur Assembleur
ash : Appel du A-shell
awk : Langage de programmation pour traitement de fichier
banner : Sortie d'une bannière
basename : Extraction du nom de fichier d'un chemin d'accès
bash : Appel du Bourne Again Shell
bc : : Interpréteur de calcul
bg : Exécuter un processus en arrière plan
break : Provoque la sortie d'une boucle (SCRIPT SHELL)
cal : Affichage d'un calendrier
case : structure de contrôle à choix multiple
cat : Afficher un fichier (texte)
catman : Mise a jour de la base apropos
cd : Changer de répertoire INTERNE
cflow : Analyse du source (Debogage)
chfn : Change les informations de finger
chgrp : Permet de changer le groupe propriétaire d'un fichier
chkconfig : Informations sur les niveaux de démarrage et les services
chmod : Change les permissions d'accès sur un fichier
chown : Permet de changer le propriétaire d'un fichier
chroot : Changement du répertoire racine d'une commande
chsh : Changement de Shell de connexion
ci : stocker la version courante (.v)
clear : Efface l'écran
clock : Retourne l'heure
cmp : Comparaison de deux fichiers
continue : Reprise d'une boucle interrompue avant son terme
consolechars : Modifie la police caractères de la console
co : Extrait la version d'un programme
cp : Copie de fichiers
cpio : Copie de fichier archive pour la sauvegarde
cpp : Compilateur C
crontab : Exécution de commandes a intervalles réguliers
ctags : Analyse du source (Débogage)
cut : Découpage de segment de ligne dans un fichier
cxref : Analyse du source (Débogage)
cvs : Gestion de projet CVS
date : Date et heure du système
dbx : Debugueur
dc : : Desk Calculator (Calculatrice en notation Polonaise Inverse)
dd : Propose une copie bloc a bloc
debugfs : Recherche d'erreurs dans un système de fichier
depmod : Générer les dépendances entre modules

df : Espace libre sur support de données
diff : Affiche les différences entre les fichiers (cf patch)
doomainname : modifie le nom de domaine NIS
dosfsck : Vérifie une partition DOS
du : Utilisation faite du disque par répertoire
dump : Sauvegarde des fichiers
dumpe2fs : Détails d'un système de fichier
dumpkeys : Dump de la table clavier
e2fsck : Vérifie une partition LINUX (ext 2)
echo : Affiche un texte
edquota : Changement des quotas d'espace disque
egrep : Recherche avec filtres étendue
emacs : Editeur de texte Emacs
env : Changement de l'environnement d'une variable
eval : Exécution multiple de commande de Shell
exit : Quitter le shell actuel INTERNE
export : Permet d'exporter une variable d'environnement
expr : Exploitation et calcul d'expressions
fdisk : Modifie la table des partitions
false : Valeur de retour standard des scripts du Shell
fc : Rappel de lignes de commandes
fdisk : Changement des partitions d'un disque dur
fdformat : Formater un périphérique
fg : Exécuter un processus au premier plan
fgrep : Recherche rapide dans un fichier
file : Déterminer le type du fichier
find : Recherche récursive de fichiers dans un répertoire
finger : Finger des utilisateurs
for : Structure de contrôle des boucles
free : Afficher la mémoire libre (RAM/swap)
fsck : Contrôle du système de fichiers
gcc : Compilateur GNU C
gdb : Debugueur GNU
gpasswd : Gestion de propriétés de groupes
gproff : Analyse de performance GNU (profilage)
groupadd : Ajout d'un groupe
groupdel : Suppression d'un groupe
groupmod : Changement des propriétés d'un groupe
groups : Affiche les groupes dont on fait partie
grpck : Vérification de syntaxe sur le fichier groupe
guile : Interpréteur LISP
gzip : Compression de fichiers
hostname : Fixe ou retourne le nom de l'ôte
hwclock : Retourne l'heure
id : Affichage no utilisateurs et groupe
ident : Extrait le numéro de version d'un exécutable
if : Décisions d'un script Shell
ifconfig : Affiche des infos sur les interfaces disponibles
ifdown : Permet d'arrêter une interface
ifup : Permet de démarrer une interface

info : Appelle les pahes infos
insmod : Chargement d'un module standalone
ipcs : Affichage d'infos sur la communication inter-process
ipfwadm : Configuration du FireWall (ipchains)
isapnp : Configuration des périphériques ISA PNP
jobs : Permet de périphériques les processus en fond
join : Jonction de deux fichiers
key : [S/KEY] Générateur de mot de passe
kill : Permet d'envoyer des signaux aux processus
ksyms : Affiche les symboles exportes par le noyau
lclint : Détection d'imperfection de programme
ldconfig : Modifie le fichier des chemins vers bibliothèques
ldd : Affiche les bibliothèques utilises par un programme
less : Affiche un texte (possibilité de déplacement)
let : Arithmétique dans le Shell
lilo : Installer un secteur d'amorçage
lint : Détection d'imperfection de programme
ln : Crée un lien vers un fichier
loadkeys : Charge une table clavier
locate : Permet de retrouver rapidement un fichier
logger : Permet d'envoyer des messages au syslogd
logname : Affichage du nom d'utilisateur
lpd : Le démon d'impression de BSD
lpq : Affichage des files d'attente d'impression
lpr : Impression de fichiers
lprm : Suppression d'un job de la file d'attente
lptest : Pour tester les imprimantes
ls : Affiche le contenu du répertoire courant
lsattr : Affichage des attributs etendus de fichiers
lsmod : Affiche les modules chargés
mail : Envoi et réception de messages électroniques
makewhatis : Cree une BD pour les mot clefs de man
man : Affichage de l'aide en ligne
mattrib : Change les attributs d'un fichier MSDOS
mcd : Change de répertoire MSDOS
mcopy : Copie un fichier MSDOS
mdel : Efface un fichier MSDOS
mdir : Affiche un répertoire MSDOS
mdu : Affiche l'espace utilise par un répertoire MSDOS
mesg : Gestion des accès sur terminaux par write et talk
mformat : Formate un système de fichier MSDOS
mkfatimage16 : Génère une image disque virtuelle pour DOSEMU
mkbootdisk : Créer une disquette de démarrage
mkboot : Créer une disquette de démarrage (Debian)
mkdir : Crée un répertoire
mkdosfs : Cree un système de fichier MSDOS
mke2fs : Cree un système de fichier linux (ext2 ou ext3)
mkfifo : Création d'un tube nommé
mkinitrd : Génère un disque mémoire
mknod : Création de fichiers de périphérique et des FIFO

mkpasswd : Cree un nouveau mot de passe
mkswap : Formate la partition de swap
mlabel : Renomme un disque MSDOS
mmbd : Serveur de noms NetBIOS (netbios-ns)
mmd : Cree un répertoire MSDOS
mmount : Monte un disque MSDOS
mmove : Monte ou renomme un répertoire MSDOS
modprobe : Chargement d'un module et de ses dépendances
more : Afficher un fichier (texte) page par page
mount : Montage système de fichier
mpartition : Partition sur un disque MSDOS
mrd : Efface un répertoire MSDOS
mren : Renomme un fichier MSDOS
mroff : Utilitaire de formatage des pages man
mtools : Commandes msdos utilisables
mtype : Affiche le contenu d'un fichier MSDOS
mv : Déplacer ou renommer un fichier
netcfg : Configuration graphique du réseau
netstat : Affiche des infos sur la config réseau
nfstat : Affiche des statistiques sur RPC
newgrp : Changer de groupe courant
nice : Modifier la priorité d'un processus (valeur entre 1 et 20)
nohup : La commande suivante ignore le signal 1
nslookup : Recherche d'adresse IP via DNS
od : Affichage des d
passwd : Changer de mot de passe
patch : Modifier le code source par versions
perl : Interpréteur PERL
pnpdump : Scanne les cartes PnP
prof : Analyseur de performance (profilage)
ps : Affiche la liste des processus
python : Interpréteur Python
pwd : Affiche le chemin courant
quotacheck : Vérifie les quotas disque
quotaon : Active les quotas disque
quotaoff : Désactive les quotas
rcs : Pour entrer des descriptions de programme
rcdiff : Différences entre deux révisions d'un programme
rdev : Modifie les paramètres de démarrage d'une disquette de boot
read : Lecture d'une valeur
readonly : Protection des variables Shell contre l'écrasement
reboot : Redémarrage
renice : Permet de modifier le facteur de priorité après lancement
return : Fin prématurée d'un Shell
rlog : Affiche un résumé des modifications du fichier
rlogin : Similaire à telnet
rm : Effacer un fichier
rmdir : Effacer un répertoire
rmmod : Déchargement d'un module
rsh : Exécute une commande sur un serveur distant

rpcinfo : Affiche des informations sur le port mappeur RPC
rusers : Liste les utilisateurs connectés
ruptime : Affiche le statut des machines locales sur le réseau
rwall : Ecrit a tous les utilisateurs du réseau local
rwho : Affiche tous les utilisateurs connectés
scanpci : Effectue un scan de tous les périphériques PCI
sdb : Debugueur
sed : Editeur de texte batch
select : Sélection de menu simple dans le Shell
set : Gestion des variables et du comportement du Shell
setserial : Configuration interface série
setsysfont : Fixe le fond de la console
sg : Changement de groupe
shift : Conversion des paramètres de position (Shell)
showkey : Affiche les scnacodes des touches clavier
shutdown : arrêt du système (-h now ou -r 2)
sleep : Interruption du traitement pendant un certain temps
smbd : Le daemon SMB (netbios-ssm)
smbclient : Un client SMB pour machines UNIX
smbprint : Un script pour imprimer sur un hote SMB
smbstatus : Liste des connexions SMB présentes
smbrun : Un script pour faciliter le lancement d'applications sur des hôtes SMB
sort : Filtre de tri ligne par ligne
startx : Démarrage du serveur X
strace : Trace l'exécution d'un programme
strip : Manipule les infos de debuggage sur un exécutable
stty : configuration d'une interface série
su : Changement d'utilisateur
sudo : Permet d'exécuter des commandes en tant que super utilisateur
sulogin : Login root en urgence
sum : calcule une somme de contrôle
suspend : Place un Shell en arrière plan
swapon : Active une partition de swap
swapoff : Désactive une partition de swap
sync : Sauvegarde de la mémoire tampon d'entrée sortie
sysctl : Affiche ou modifie les variables système
tac : Affiche un fichier dans l'ordre inverse
tail : Affichage de la dernière partie d'un fichier
talk : Envoi de messages inter utilisateurs
tar : Sauvegarde et archivage de fichiers
tcsh : Appel du Tene C-Shell
tclsh : Interpréteur TCL
tee : Redirection de saisie
telint : Gestion des "niveaux" système
test : Contrôle de conditions
time : Permet de mesurer le temps nécessaire à une commande
tin : Lecteur de news
touch : Permet de créer un fichier vide
tr : Conversion de caractères
traceroute : Reconstitue la route qu'emprunte un paquet

trap : Gestion des signaux (Shell)
true : Valeur de retour standard pour un script Shell
tty : Affichage des noms des terminaux
tune2fs : Modifie les options sur le système de fichier
type : Affiche le type du fichier
ulimit : Pour définir la taille maximum d'un fichier
umask : Définit les droits (négatifs) de création d'un fichier
umount : Démontage système de fichiers
umssync : Synchronisation d'un système de fichier UMSDOS
unalias : Suppression d'un nom d'alias
uname : Informations (processeur , version) système
unset : Suppression de définitions de variables et de fonctions
until : Structure de contrôle des boucles
update : Remet a jour les super blocs du système de fichiers
updatedb : Reconstruit la BD des positions de fichiers
uptime : Affiche l'uptime de la machine
useradd : Ajout d'un nouvel utilisateur
userdel : Suppression d'un utilisateur
usermod : Changement des attributs d'un utilisateur
vi : Editeur texte vi
vipw : Editeur pour le fichier password
wait : Attend la fin d'un processus en arrière plan
wall : Envoi d'un message à tous les utilisateurs
wc : comptabilisation des caractères, des mots et des lignes
whatis : Retourne une brève description de la fonction
which : Indique quel fichier sera appelé (dans PATH)
while : Structure de contrôle de boucle
who : Affiche les utilisateurs courants
write : Envoi d'un message à d'autres utilisateurs
xargs : combinaison de lignes de commandes et de saisie clavier
ypinit : Génère les maps NIS
ypmatch : Génère les maps NIS
yppasswd : Modifie le mot de passe NIS
ypwhich : Permet de savoir le nom du serveur NIS
whereis : Permet de recherche un exécutable dans le PATH
winkey : [LogDaemon] Générateur de mot de passe