**Linux**

C'est un système d'exploitation:

 Il fait le lien entre votre matériel (carte graphique, mémoire, imprimante) et vos logiciels.

Un système d'exploitation se dit *Operating System* en anglais, que l'on abrège en « OS ». J'utiliserai souvent cette abréviation par la suite, ne soyez donc pas surpris !

Si on résume l'ordre des choses, au final nous avons donc cela :

1. écran de boot ;
2. démarrage du système d'exploitation (Windows) ;
3. lancement des programmes (jeux, Internet, mail…).

Beaucoup de gens croient qu'il faut faire un choix : Linux ou Windows. Rien n'est plus faux : vous pouvez très bien avoir deux OS (ou plus !) installés sur votre ordinateur.

Dans ce cas, juste après l'écran de boot, vous aurez un programme appelé Boot Loader qui s'affichera pour que vous puissiez faire votre choix. Le boot loader dont nous parlerons ici porte le doux nom de GRUB.

Les programmes Windows ne fonctionnent pas sous Linux et inversement.

les programmes sous Linux ont d'énormes avantages :

* ils sont gratuits : vous verrez que sous Linux, presque tous les programmes sont gratuits ;
* les logiciels sont souvent mis à jour et ce toujours gratuitement ! Vous verrez donc fréquemment vos logiciels préférés évoluer et n'aurez pas à payer trois cents euros pour vous mettre à jour !
* certains de ces logiciels sont meilleurs que ceux que l'on trouve sous Windows. D'ailleurs, certains n'existent même pas sous Windows ! Vous découvrirez de nouvelles fonctionnalités et finirez par gagner du temps tout en utilisant plus efficacement votre ordinateur.

#### L'informatique en 1984

Nous sommes donc en 1984. À cette époque, l'informatique n'est pas très développée. Microsoft vient de sortir son premier OS : MS-DOS. Mais ce dernier est encore loin d'être abouti.

Mais MS-DOS était-il le seul OS existant à l'époque ?

Non ! Il y en avait d'autres mais bien moins connus du grand public.  
Celui qui était considéré comme le meilleur s'appelait « Unix ». Il était beaucoup plus puissant que MS-DOS mais aussi plus compliqué à utiliser, ce qui explique pourquoi seuls les informaticiens professionnels l'utilisaient.  
Il est aussi beaucoup plus ancien : ses origines remontent à 1969 !

Graphiquement, Unix ressemblait beaucoup à MS-DOS : du texte blanc sur un fond noir. Il faut dire qu'à l'époque les ordinateurs n'étaient pas vraiment capables de faire mieux.

C'est justement à cette époque, en 1984, que Richard Stallman créa le projet GNU.  
Richard Stallman était alors chercheur en intelligence artificielle au MIT. Il voulait créer un nouveau système d'exploitation fonctionnant comme Unix (les commandes restant les mêmes).

Pourquoi vouloir créer une « copie » d'Unix ?  
Parce qu'Unix était payant et devenait de plus en plus cher ! Richard Stallman a voulu réagir en proposant une alternative gratuite : le projet GNU était né.

Bon à savoir : Mac OS X est lui aussi basé sur Unix. En revanche, MS-DOS et Windows sont complètement à part.

##### GNU est un système d'exploitation libre

GNU ne devait pas seulement être un OS gratuit ; il devait également être « libre ».

Quelle différence ?

Un programme libre est un programme dont on peut avoir le code source, c'est-à-dire la « recette de fabrication ».  
Au contraire, Windows est un OS propriétaire dont le code source est conservé par Microsoft. Imaginez que c'est un peu comme le Coca-Cola : personne ne connaît la recette de fabrication (il y a bien des gens qui essaient de l'imiter, mais bon…). On ne peut donc pas le modifier ou regarder comment il fonctionne à l'intérieur.

On dit aussi souvent que le programme est « Open Source », car son code source est ouvert ; tout le monde peut le voir.  
Il existe quelques légères différences entre un programme « Open Source » et un programme « libre », mais nous n'entrerons pas dans les détails ici.

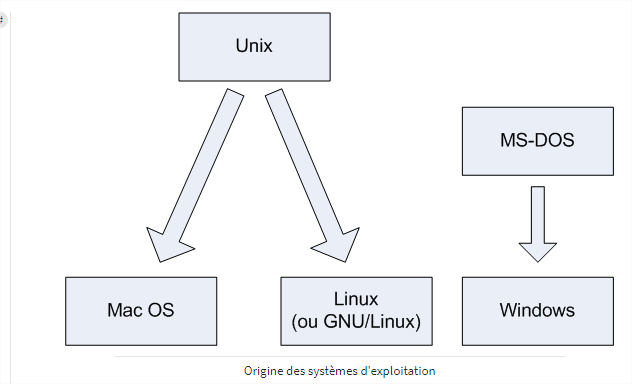
#### Pendant ce temps, Linus Torvalds s'amusait

En 1991, Linus Torvalds, un étudiant de l'Université de Helsinki (Finlande), entreprend de créer sur son temps libre son propre système d'exploitation.  
Ce système a pris le nom de Linux, en référence au nom de son créateur (Linux est la contraction de Linus et Unix).

Quel rapport avec GNU ? Eh bien il se trouve que ces deux projets étaient complémentaires : tandis que Richard Stallman créait les programmes de base (programme de copie de fichier, de suppression de fichier, éditeur de texte), Linus s'était lancé dans la création du « cœur » d'un système d'exploitation : le noyau.

Le projet GNU (programmes libres) et Linux (noyau d'OS) ont fusionné pour créer GNU/Linux.

Théoriquement, on doit donc parler de GNU/Linux. C'est toutefois un peu difficile à écrire et prononcer, et par abus de langage, on dit souvent juste « Linux ». C'est donc pour cela que je continuerai à parler de « Linux » dans la suite du livre, même si le nom *politiquement correct*est « GNU/Linux » puisqu'il s'agit de la fusion de deux projets complémentaires.



Les programmes Linux n'utilisent pas du tout le même code source que ceux d'Unix (celui-ci était d'ailleurs propriétaire, donc privé). Ils ont été complètement réécrits mais fonctionnent de la même manière.

Les différentes distributions de Linux

Voici ce qui peut différer d'une distribution à l'autre :

* l'installation : elle peut être très simplifiée comme très compliquée ;
* la gestion de l'installation des programmes. Si elle est bien faite et centralisée, elle peut rendre l'installation de nouveaux logiciels plus simple que sous Windows, comme nous le verrons plus loin !
* les programmes préinstallés sur l'ordinateur (Windows est par exemple livré avec Internet Explorer et Windows Media Player).

En fait, une distribution est en quelque sorte l'emballage de Linux. Le cœur, lui, reste le même sur toutes les distributions.

Quelle que soit la distribution que vous installez, vous obtenez un Linux compatible avec les autres. Certaines distributions sont juste plus ou moins faciles à prendre en main.

#### Les différentes distributions existantes

Il existe un grand nombre de distributions Linux différentes, mais voici au moins les principales :

* Slackware : une des plus anciennes distributions de Linux. Elle existe encore aujourd'hui !
* Mandriva : éditée par une entreprise française, elle se veut simple d'utilisation ;
* Red Hat : éditée par une entreprise américaine, cette distribution est célèbre et très répandue, notamment sur les serveurs ;
* SuSE : éditée par l'entreprise Novell ;
* Debian : la seule distribution qui soit gérée par des développeurs indépendants plutôt que par une entreprise. C'est une des distributions les plus populaires.

Debian a tellement de succès que de nombreuses distributions sont basées sur Debian :

* Knoppix ;
* Skolelinux ;
* Ubuntu ;
* …

Ce sont donc des… distributions de distributions. :-)

Logo de Ubuntu:



Quelle que soit la distribution il existe deux façons d'utiliser Linux :

* en mode console (équivalent à DOS) ;
* en mode graphique (équivalent à Windows).

#### En mode graphique

Il y a plusieurs modes graphiques. Tous les modes graphiques sont basés sur un programme appelé X (voilà un nom court et facile à retenir). X est en fait la brique de base du mode graphique sous Linux.

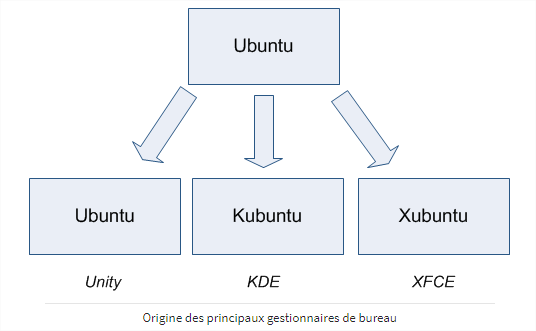
Par-dessus X vient se greffer un programme appelé le gestionnaire de bureau. Le rôle du gestionnaire de bureau est de gérer les fenêtres, leur apparence, leurs options, etc.

Le concept de gestionnaire de bureau n'existant pas sous Windows, il s’agit donc de quelque chose nouveau pour vous.  
Certes, sous Windows on peut changer l'apparence (le « skin »), mais cela s'arrête là. Le bureau reste le même, il y a toujours une barre des tâches avec le menu *Démarrer*, toujours un *Poste de Travail*, etc.  
Sous Linux en revanche le fonctionnement peut radicalement changer d'un gestionnaire de bureau à l'autre !

##### Présentation des principaux gestionnaires de bureau

Je ne vais pas vous faire la liste de tous les gestionnaires de bureau qui existent (ils sont vraiment nombreux !), mais commencez déjà par retenir ces quatre-là, qui sont probablement les plus célèbres :

* Unity ;
* Gnome ;
* KDE ;
* XFCE.



Il est possible d'installer Ubuntu directement depuis Windows, mais la manipulation classique consiste plutôt à redémarrer l'ordinateur avec le CD d'Ubuntu dans le lecteur.

* La première fois, Ubuntu se charge directement sur le CD, en mode Live CD. Votre disque dur n'est pas modifié et vous pouvez tester Ubuntu tranquillement.

#### Les systèmes de fichiers

Sur chaque partition, les fichiers sont organisés selon ce qu'on appelle un système de fichiers. C'est en quelque sorte une façon d'organiser les fichiers : ils sont tous référencés dans une sorte d'annuaire gigantesque.  
Le système de fichiers permet aussi de dire qui a le droit de voir tel ou tel fichier. D'autre part, les systèmes de fichiers récents sont dits « journalisés », c'est-à-dire qu'en cas de crash (votre PC est éteint brusquement), le système est capable de retrouver ses fichiers sans trop de problèmes.

Règle importante : il ne peut y avoir qu'un seul système de fichiers par partition.

Il existe un nombre important de systèmes de fichiers différents, en voici quelques-uns à connaître.

##### Systèmes de fichiers Microsoft (DOS et Windows)

* FAT 16 : un très vieux système de fichiers, capable de gérer jusqu'à 4 Go de données. Il est donc impossible de faire une partition en FAT 16 de plus de 4 Go. Il était très utilisé à l'époque du DOS et aux débuts de Windows 95.
* FAT 32 : une évolution du FAT 16, qui pousse la limite de taille à 2 To (2 000 Go).
* NTFS : apparu avec Windows NT, puis réutilisé par Windows XP, il permet de créer des partitions d'une taille allant jusqu'à 16 Eo (16 Exaoctets, soit 16 000 000 000 de Gigaoctets). D'autre part, on peut donner des droits sur certains fichiers, les crypter, les compresser, etc. Enfin, les fichiers sont censés moins se fragmenter… cependant, ils se fragmentent toujours.

##### Systèmes de fichiers Linux

* + ext2 : c'est le système de fichiers qui a longtemps été utilisé sous Linux. Il a été développé par un français (Rémy Card) et présente la particularité de très peu se fragmenter. Ainsi, sous Linux et depuis longtemps, il n'y a pas besoin de faire de défragmentation.
  + ext3 : l'ext3 est très proche de l'ext2, à une différence majeure près : la journalisation. En effet, ext2 n'était pas journalisé et en cas de crash du disque, on risquait plus facilement une perte de données. Ce n'est plus le cas avec l'ext3.

À noter que l'ext2 et l'ext3 sont parfaitement compatibles entre eux, dans un sens comme dans l'autre.

* ext4 : une amélioration de l'ext3, relativement récente, qui améliore la prise en charge des gros disques durs et diminue les problèmes de fragmentation des fichiers.

##### À propos du nom des disques

Si vous avez un seul disque dur sur votre ordinateur, pas de problème, vous ne risquez pas de vous tromper. Toutefois, si par hasard vous avez plusieurs disques, je pense que vous apprécierez que je vous explique comment les disques durs sont nommés sous Linux.

En effet, c'est très différent de Windows où l'on a l'habitude des sempiternels C:,D:,E:, etc.  
On va découvrir les noms des disques sous Linux avec un exemple : hda.

* h : la première lettre indique si le disque est de type IDE ou SCSI (un type de connexion différent à la carte mère). S'il est de type IDE, la lettre est un « h », si c'est un SCSI (ou un S-ATA), la lettre est un « s ».
* d : cette lettre ne change pas.
* a : c'est cette lettre qui indique les différents disques durs. hda représente le premier disque dur IDE, hdb le second, hdc le troisième, etc.

Lorsque l'on crée des partitions, on ajoute généralement un chiffre représentant le numéro de la partition. Ainsi, si on a trois partitions sur notre disque hda, elles seront nommées hda1, hda2, hda3…

Un disque dur peut être virtuellement découpé en plusieurs sections différentes : c'est le principe du partitionnement.

* Pour installer Linux, vous devez modifier le partitionnement de votre disque. L'opération consiste à réduire l'espace alloué à Windows pour faire de la place à Linux.
* Les partitions Windows stockent les fichiers selon un système appelé NTFS, tandis que les partitions sous Linux utilisent plus souvent les systèmes ext3 et ext4.
* On crée une partition spéciale appelée swap d'environ 1 Go, qui sert d'extension à la mémoire vive lorsque celle-ci est pleine.
* Après l'installation, un outil appelé GRUB vous demandera à chaque démarrage quel système d'exploitation vous souhaitez lancer.



Installez Linux dans une machine virtuelle

La virtualisation est une technique de plus en plus répandue en informatique : cela consiste à faire tourner un ordinateur « virtuel » dans votre ordinateur.

VirtualBox est un outil qui permet de faire tourner un ordinateur virtuel au sein d'un système d'exploitation.

**Découverte du bureau Unity**

**Nautilus, l'explorateur de fichiers**

Nautilus est un explorateur de fichiers. C'est un programme du même type que l'explorateur de Windows.

**La console**

La plupart des commandes de la console de Linux sont des « copies » d’Unix

Voici les raccourcis à connaître pour accéder à la console :

* Ctrl + Alt + F1 : terminal 1 (tty1) ;
* Ctrl + Alt + F2 : terminal 2 (tty2) ;
* Ctrl + Alt + F3 : terminal 3 (tty3) ;
* Ctrl + Alt + F4 : terminal 4 (tty4) ;
* Ctrl + Alt + F5 : terminal 5 (tty5) ;
* Ctrl + Alt + F6 : terminal 6 (tty6) ;
* Ctrl + Alt + F7 : retour au mode graphique

**La console en mode graphique**

C’est le fait d’avoir un terminal ouvert dans une petite fenêtre tout en ayant en arrière plan le bureau en mode graphique.

Pour KDE, il suffit de rechercher "Konsole" dans le lanceur

Pour Unity, il suffit de rechercher "Terminal" dans le lanceur

Il est possible d’ouvrir plusieurs terminaux en même temps>> Files>>Open a terminal

**L'accès à distance en SSH avec PuTTY**

Une des grosses forces de la console, c'est d'être accessible à distance par Internet. Il suffit que votre machine soit connectée au réseau pour que vous puissiez vous logger de n'importe quel ordinateur dans le monde et faire comme si vous étiez chez vous !

Cela permet de lancer l'exécution d'un programme pour qu'il soit prêt lorsque vous serez rentrés chez vous… mais surtout, c'est comme ça que l'on administre un serveur sous Linux.

Un serveur est – pour faire simple – un ordinateur tout le temps connecté à Internet. Il permet d'offrir des services divers et variés aux internautes. Par exemple, il y a des serveurs web dont le rôle est de… distribuer des pages web.

La grande majorité des serveurs tourne sous Linux. Lorsque vous allez sur un site, il y a de très fortes chances pour que ce soit un serveur Linux qui vous réponde. Les serveurs Windows existent aussi, mais ils sont plus rares et on apprécie en général la stabilité de Linux ainsi que la possibilité de l'administrer à distance en ligne de commande.

**Telnet et SSH**

Pour communiquer entre votre ordinateur et le serveur, il faut un **protocole**. C'est un ensemble de règles pour que deux ordinateurs puissent discuter entre eux.

Il existe des tonnes et des tonnes de protocoles pour communiquer par Internet, mais pour ce qui est d'accéder à la ligne de commande à distance, c'est-à-dire à la console, il y en a deux principaux.

* **Telnet** : le protocole le plus basique, qui présente le gros défaut de ne pas crypter les données échangées entre vous et le serveur. Si un pirate « écoute » vos échanges par un moyen ou un autre, il pourrait récupérer des informations sensibles, en particulier votre mot de passe lorsque vous l'envoyez à la connexion. Ce moyen de connexion reste utilisé mais peu par rapport à SSH.
* **SSH** : c'est de très loin le protocole le plus utilisé (et que l'on préfère) car il permet de crypter les données et de sécuriser ainsi la connexion avec le serveur.

**PuTTY**

Pour accéder à distance à un ordinateur sous Linux connecté à Internet, vous avez besoin d'un programme spécial capable de restituer la ligne de commande à distance.

Il existe plusieurs programmes capables de se connecter en SSH à un serveur Linux. Le plus célèbre sous Windows est sûrement PuTTY

Dans l’invite de commande, le symbole tout à la fin indique sous quel statut on est loggué : utilisateur normal ou admin

* **$** : signifie que vous êtes en train d'utiliser un compte utilisateur « normal », avec des droits limités (il ne peut pas modifier les fichiers système les plus importants).
* **#** : signifie que vous êtes en mode superutilisateur, c'est-à-dire que vous êtes connectés sous le pseudonyme « root ».

**Commandes et paramètres**

Le manuel d’utilisation est un outil de référence, là où l'on peut trouver la réponse à TOUTES ses questions

Pour information, il est courant de voir un linuxien dire « RTFM » aux débutants qui posent des questions qu'il estime « simples ». RTFM est l'abréviation de « Read the fucking manual! »

**Les commandes simples**

Date

ls

C’est l'abréviation de « list », qui signifie « **lister les fichiers et dossiers du répertoire actuel** ».

Une commande est constituée d'un mot et ne contient aucun espace

**Les paramètres**

Les paramètres sont des options que l'on écrit à la suite de la commande. La commande et les paramètres sont séparés par un espace

Les paramètres les plus courants sont constitués d'une seule lettre précédée d'un tiret. Par exemple :

commande -d

Si on doit donner plusieurs paramètres, on peut faire comme ceci :

commande -d -a -h ou commande -dah

Les paramètres constitués de plusieurs lettres sont précédés de deux tirets, comme ceci :

commande --parametre

Cette fois, pas le choix : si vous voulez mettre plusieurs paramètres longs, il faudra ajouter un espace entre chacun d'eux :

commande --parametre1 --parametre2

On peut aussi combiner les paramètres longs et les paramètres courts dans une commande :

commande -daUh --autreparametre

**Les valeurs des paramètres**

Certains paramètres nécessitent que vous les complétiez avec une valeur.

Avec un paramètre court :

commande -p 14

« **Je veux voir la liste de tous les fichiers de plus de 14 Mo** »

Si c'est un paramètre long, on fait en général comme ceci :

commande --parametre=14

Certains paramètres sont donc un peu différents et dépendent vraiment des commandes. Par exemple avec ls, si on ajoute le nom d'un dossier (ou sous-dossier), cela affichera le contenu de ce dossier au lieu du contenu du dossier courant

**Retrouver une commande**

**Autocomplétion de commande**

Si tu oublies une commande mais te souviens des premières lettres, tu les tapes, puis tu tapes deux fois sur tabulation.

Ainsi l’ordinateur va te lister toutes les commandes qui commencent par exemple par da soit date et dash

Plus sympa encore, s'il n'y a qu'un seul résultat correspondant à votre recherche, l'ordinateur complètera avec les lettres qui manquent et vous n'aurez plus qu'à taper sur Entrée !

Si tu tapes deux fois sur tabulation sans rien écrire, l’ordinateur te proposera de lister toutes les commandes Linux disponibles et ce en fonction du nombres de programmes installés, ça peut être plus de 2000.

Certains Linux veulent que tu rentres au moins une lettre pour t’afficher toutes les commandes commençant par cette lettre.

**L'historique des commandes**

On peut le voir en tapant sur la touche de la flèche du haut ou bien avec la commande history

On peut rechercher une commande tapée avec Ctrl + R

**Quelques raccourcis clavier pratiques**

**Ctrl + L** ou clear : efface le contenu de la console.

Exit pour quitter la console.

* **Ctrl + A** : ramène le curseur au début de la commande. La touche Origine a le même effet (elle est située à côté de la touche Fin et représentée par une flèche pointant en haut à gauche).
* **Ctrl + E** : ramène le curseur à la fin de la ligne de commandes. La touche Fin a le même effet.
* **Ctrl + W** : supprime le premier mot situé à gauche du curseur. Un « mot » est séparé par des espaces ; on s'en sert en général pour supprimer le paramètre situé à gauche du curseur.
* **Ctrl + Y** : si vous avez supprimé du texte avec une des commandes Ctrl + U, Ctrl + K ou Ctrl + W qu'on vient de voir, alors le raccourci Ctrl + Y « collera » le texte que vous venez de supprimer. C'est donc un peu comme un couper-coller.

**La structure des dossiers et fichiers**

Au lieu de séparer chaque disque dur, lecteur CD, lecteur de disquettes, lecteur de carte mémoire… Linux place en gros tout au même endroit.

**Deux types de fichiers**

Pour faire simple, il existe deux grands types de fichiers sous Linux :

* **les fichiers classiques** : ce sont les fichiers que vous connaissez, ça comprend les fichiers texte (.txt,.doc,.odt…), les sons (.wav,.mp3,.ogg), mais aussi les programmes. Bref, tout ça, ce sont des fichiers que vous connaissez et que vous retrouvez dans Windows ;
* **les fichiers spéciaux** : certains autres fichiers sont spéciaux car ils **représentent** quelque chose. Par exemple, votre lecteur CD est un fichier pour Linux. Là où Windows fait la distinction entre ce qui est un fichier et ce qui ne l'est pas, Linux, lui, dit que **tout est un fichier**.

**La racine**

Dans un système de fichiers, il y a toujours ce qu'on appelle une racine, c'est-à-dire un « **gros dossier de base qui contient tous les autres dossiers et fichiers** ».

Sous Windows, il y a en fait plusieurs racines.C:\est la racine de votre disque dur,D:\est la racine de votre lecteur CD (par exemple).

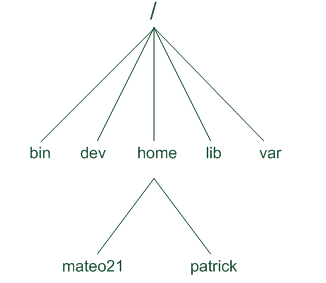
Sous Linux, **il n'y a qu'une et une seule racine** : « / ». Comme vous le voyez, il n'y a pas de lettre de lecteur car justement, Linux ne donne pas de nom aux lecteurs comme le fait Windows. Il dit juste « **La base, c'est/** ».

**Quelques dossiers de la racine**

* **bin** : contient des programmes (exécutables) susceptibles d'être utilisés par tous les utilisateurs de la machine.
* **boot** : fichiers permettant le démarrage de Linux.
* **dev** : fichiers contenant les périphériques. On y retrouve ainsi par exemple le fichier qui représente le lecteur CD.
* **etc** : fichiers de configuration.
* **home** : répertoires personnels des utilisateurs à la manière du dossierMes documents de Windows.

**root** : c'est le dossier personnel de l'utilisateur « root ».

**var** : ce dossier contient des données « variables », souvent des logs (traces écrites de ce qui s'est passé récemment sur l’ordinateur).



D’autres commandes :

pwd : affiche le dossier actuel, c’est l'abréviation de « Print Working Directory »

Chaque commande sous Linux correspond à un programme que l’on peut appeler n’importe où et n’importe quand.

which : Permet de savoir dans quel dossier se trouve une commande

La commande which prend un paramètre : le nom de la commande dont vous voulez connaître l'emplacement.

Paramètres à la commande ls :

**-a : afficher tous les fichiers et dossiers cachés**

**-l : liste détaillée**

**-h : afficher la taille en Ko, Mo, Go…**

**-t : trier par date de dernière modification**

On peut combiner un peu tous les paramètres, ce qui donne ls -larth

Plutôt que d'avoir à réécrire ls -larth à chaque fois (c'est un peu long), on peut créer un alias, c'est-à-dire une commande synonyme. Par exemple, j'ai créé l’alias ll(deux fois « L ») qui est automatiquement transformé par Linux en ls -larth

cd ne prend pas plein de paramètres mais juste un seul : le nom du dossier dans lequel vous souhaitez aller.

Si on veut aller à la racine, il suffit de taper cd /

Après avoir tapé cd /, on se retrouve à la racine. L'invite de commandes a changé et le~a été remplacé par un/

Pour revenir au dossier précédent on tape : cd ..

Si on avait voulu reculer de deux dossiers parents, on aurait écrit : cd ../..

Il y a en fait deux façons de changer de dossier : en indiquant un **chemin relatif**, ou en indiquant un **chemin absolu**.

Les chemins relatifs

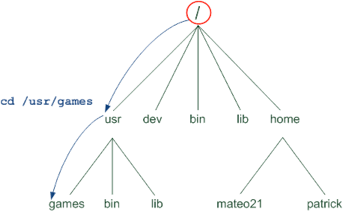
Un chemin relatif est un chemin qui dépend du dossier dans lequel vous vous trouvez.

Si tu as un sous-dossier villes dans un dossier pays et que tu te trouves dans le dossier pays, tu n'auras qu'à taper : cd villes

Les chemins absolus

Contrairement aux chemins relatifs, les chemins absolus fonctionnent quel que soit le dossier dans lequel on se trouve.

Un chemin absolu est facile à reconnaître : il commence toujours par la racine (/). Vous devez ensuite faire la liste des dossiers dans lesquels vous voulez entrer. Par exemple, supposons que je sois dans/home/mateo21 et que je souhaite aller dans/usr/games. Avec un chemin absolu :

cd /usr/games

Si on avait voulu faire la même chose à coup de chemin relatif, il aurait fallu écrire :

cd ../../usr/games/

Ce qui signifie « **reviens en arrière (donc dans**/home**) puis reviens en arrière (donc dans**/**), puis va en avant dans** usr**, puis va en avant dans** games »

Retour au répertoire home :

En utilisant l'alias~

cd ~ ou uniquement cd

Autocomplétion du chemin

L'autocomplétion de chemin fonctionne de la même manière que l'autocomplétion de commande qu'on a vue dans le chapitre précédent : avec la touche Tab(Tabulation). Commencez par vous placer dans/usr :

Si par exemple tu as un dossier games dans ce dossier, tape ensuite juste cd ga, puis appuie sur Tab. C'est magique, le nom du dossier a été automatiquement complété !

du: taille occupée par les dossiers

La commande « du », pour *Disk Usage* (utilisation du disque) vous donne des informations sur la taille qu'occupent les dossiers sur votre disque.

Il faut juste taper du

Si tu tapes du -h il t'affichera les valeurs de manières correctes

-a : afficher la taille des dossiers ET des fichiers

-s : avoir juste le grand total

* 1. Manipuler les fichiers

Cat : afficher tout le fichier

La commandecatpermet d'afficher tout le contenu d'un fichier dans la console d'un coup.

le paramètre-nqui permet d'afficher les numéros de ligne :

cat -n syslog

less : afficher le fichier page par page

Tapez q pour sortir de l'aide.

head : afficher le début du fichier

head -n 3 // Va afficher les 3 premières lignes :

mateo21@mateo21-desktop:/var/log$ head -n 3 syslog

Nov 14 00:44:23 mateo21-desktop syslogd 1.4.1#21ubuntu3: restart.

Nov 14 00:44:23 mateo21-desktop anacron[6725]: Job `cron.daily' terminated

Nov 14 00:44:23 mateo21-desktop anacron[6725]: Normal exit (1 job run)

tail : afficher la fin du fichier

À connaître : la combinaison de touchesCtrl + Cest utilisable dans la plupart des programmes console pour demander leur arrêt. C'est un peu l'équivalent duAlt + F4de Windows.

tail -f est une de mes commandes préférées sous Linux. C'est un bon moyen de surveiller ce qui se passe en temps réel sur un ordinateur (si vous êtes assez rapides pour suivre).

Par exemple, les logs Apache du Site du Zéro permettent de voir en temps réel qui se connecte sur le site, avec quelle IP, quel fichier a été chargé, à quelle heure, etc.  
Aux heures d'affluence du site, ce fichier évolue tellement vite qu'il est pratiquement impossible de le suivre pour un humain.

touch & mkdir : créer des fichiers et dossiers

touch : créer un fichier

En fait, il n'existe aucune commande spécialement faite pour créer un fichier vide sous Linux (ce n'est pas très utile). En général, on se contente d'ouvrir un éditeur de texte et d'enregistrer, ce qui provoque la création d'un fichier comme sous Windows.

La commande touch est à la base faite pour modifier la date de dernière modification d'un fichier. D'où son nom : on « touche » le fichier pour faire croire à l'ordinateur qu'on vient de le modifier alors que l'on n'a rien changé. Ça peut se révéler utile dans certains cas précis qu'on ne verra pas ici.

L'intérêt de touch pour nous dans ce chapitre, c'est que si le fichier n'existe pas, il sera créé ! On peut donc **aussi** utiliser touch pour créer des fichiers, même s'il n'a pas vraiment été fait pour ça à la base.

touch pays // Va créer le fichier pays

On peut créer plusieurs fichiers en une seule commande. Il vous suffit de les lister l'un après l'autre, séparés par des espaces.

touch pays villes

mkdir : créer un dossier

mkdir mondossier

On peut créer deux dossiers (ou plus !) en même temps en les séparant là aussi par des espaces :

Il y a un paramètre utile avec mkdir :-p. Il sert à créer tous les dossiers intermédiaires. Par exemple :

mkdir -p animaux/vertebres/chat

créera le dossier animaux, puis à l'intérieur le sous-dossiervertebres, puis à l'intérieur encore le sous-dossierchat !

cp & mv : copier et déplacer un fichier

cp : copier un fichier

La commande cp (abréviation de *CoPy*) vous permet comme son nom l'indique de copier un fichier… mais aussi de copier plusieurs fichiers à la fois, et même de copier des dossiers !

cp capitales France

Là il va copier le fichier capitales dans le fichier France, si France n'existe pas, il va le créer.

Copier dans un autre dossier

Avec cette commande par exemple :

cp villes maroc

Là ça va copier le fichier villes dans le dossier maroc qui existe déjà.

Si dans le dossier maroc il y a déjà un fichier nommé villes, ce dernier sera écrasé.

Si vous voulez copier un fichier dans un dossier sous un autre nom on fait :

cp villes maroc/cities

cp fichierbidon /var/log/ >> va créer le fichier dans le dossier log

Copier des dossiers

Avec l'option-R(un « R » majuscule !), vous pouvez copier un dossier, ainsi que tous les sous-dossiers et fichiers qu'il contient !

cp -R animaux autresanimaux

cela aura pour effet de copieranimauxainsi que tous ses sous-dossiers sous le nom autresanimaux mais toujours dans le même dossier.

Utiliser le joker\*

Le symbole\*est appelé *joker*, ou encore **wildcard** en anglais sous Linux.  
Il vous permet de copier par exemple tous les fichiers image.jpgdans un sous-dossier :

cp \*.jpg mondossier/

Vous pouvez aussi vous en servir pour copier tous les fichiers dont le nom commence par « so » :

cp so\* mondossier/

mv : déplacer un fichier

Très proche decp, la commandemv(*MoVe*, « déplacer » en anglais) a en fait deux utilités :

* + - * déplacer un fichier (ou un dossier) ;
      * renommer un fichier (ou un dossier).

La commandemvs'utilise pratiquement commecp :

mv fichierbidon mondossier/

Mais au lieu de copier le ficher, elle va le déplacer.

Vous pouvez aussi utiliser les jokers :

mv \*.jpg mondossier/

Renommer un fichier

La command emv permet de faire quelque chose d'assez étonnant : renommer un fichier. En effet, il n'existe pas de commande spéciale pour renommer un fichier en console sous Linux, c'est la commandemvqui est utilisée pour ça.

Par exemple :

mv fichierbidon superfichier

renommera fichierbidon en superfichier. Après cette commande,fichierbidon n'existe plus, il a été renommé.

Déplacer et renommer un fichier à la fois

Vous pouvez aussi déplacerfichierbidondansmondossiertout en lui affectant un nouveau nom :

mv fichierbidon mondossier/superfichier

rm : supprimer des fichiers et dossiers

Il n'existe pas de corbeille dans la console de Linux : le fichier est directement supprimé sans possibilité de récupération !

La commande rm(pour *ReMove*, « supprimer » en anglais) peut supprimer un fichier, plusieurs fichiers, des dossiers, voire même votre ordinateur entier si vous le voulez.

Il faut donc l'utiliser avec précaution.

rm Torcy

Vous pouvez aussi supprimer plusieurs fichiers en séparant leurs noms par des espaces :

rm fichierbidon fichiercopie

-i : demander confirmation

La commande-ipermet de vous demander une confirmation pour chacun des fichiers :

mateo21@mateo21-desktop:~$ rm -i fichierbidon

rm: détruire fichier régulier vide `fichierbidon'?

-f : forcer la suppression, quoi qu'il arrive

-v : dis-moi ce que tu fais, petit cachotier

Le paramètre-v(*Verbose*, verbeux en anglais, c'est-à-dire « parler beaucoup ») est un paramètre que l'on retrouve dans beaucoup de commandes sous Linux. Il permet de demander à la commande de dire ce qu'elle est en train de faire.

Comme vous l'avez vu, par défaut la commandermest silencieuse. Si vous supprimez de très nombreux fichiers, ça peut prendre du temps. Pour éviter que vous vous impatientiez, pensez à utiliser-v :

mateo21@mateo21-desktop:~$ rm -v fichierbidon fichiercopie

détruit `fichierbidon'

détruit `fichiercopie'

-r : supprimer un dossier et son contenu

Le paramètre-rpeut être utilisé pour supprimer un dossier (au lieu d'un fichier) ainsi que tout ce qu'il contient : fichiers et dossiers !

Notez qu'il existe aussi la commandermdir. La grosse différence avecrm -r, c'est quermdirne peut supprimer un dossier que s'il est vide ! Il faudra y avoir fait le ménage auparavant.

rm et le joker de la mort

rm -rf /\*

En clair, cette commande supprime tout votre disque dur depuis la racine, sous-dossiers compris, et ne demande aucune confirmation. Aucune possibilité de récupération, votre PC est foutu. Vous êtes bons pour une réinstallation de Linux, **et aussi de Windows** si la partition de Windows était accessible depuis Linux.

rm -rf \*

supprime tous les fichiers et sous-dossiers du dossier dans lequel je me trouve.

ln : créer des liens entre fichiers

La commande ln permet de créer des liens entre des fichiers, c'est-à-dire qu'elle permet de **créer des raccourcis**.

Il y a deux types de liens :

des liens **physiques** ;

des liens **symboliques**.

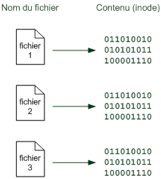
Le stockage des fichiers sur Linux.

Sur le disque dur, chaque fichier est grosso-modo séparé en deux parties :

* + - * son nom ;
      * son contenu.

Vous avez bien lu : la liste des noms de fichiers est stockée à un autre endroit que leur contenu. Cette séparation aide Linux à s'organiser.

En pratique, il y en fait trois parties : le nom, les informations de gestion (droits d'accès) et le contenu.

Chaque contenu de fichier se voit attribuer un numéro d'identification appelé *inode* (figure suivante). Chaque nom de fichier est donc associé à un inode (son contenu).

Un lien physique permet d'avoir deux noms de fichiers qui partagent exactement le même contenu, c'est-à-dire le même inode (figure suivante).

On crée un dossier puis un fichier one dans ce fichier et ensuite on va créer un lien physique : nous allons créer un two qui partagera le même inode (le même contenu) que one. On tape :

ln one two

Si vous listez les fichiers du répertoire, vous avez l'impression d'avoir deux fichiers différents :

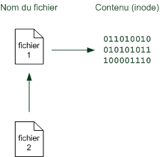
-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 one

-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 two

Si vous supprimez un des deux fichiers, l'autre fichier reste en place et le contenu sera toujours présent sur le disque. L'inode est supprimé uniquement quand plus aucun nom de fichier ne pointe dessus.

Créer des liens symboliques

Les liens symboliques ressemblent plus aux « raccourcis » dont vous avez peut-être l'habitude sous Windows. La plupart du temps, on crée des liens symboliques sous Linux pour faire un raccourci, et non des liens physiques qui sont un peu particuliers.

Le principe du lien symbolique est que l'on crée un lien vers un autre nom de fichier. Cette fois, on pointe vers le nom de fichier et non vers l'inode directement (figure suivante).

Pour créer un lien symbolique il faut utiliser le paramètre -s

ln -s one two

Cette fois, la commande détaillée ls -l sera beaucoup plus précise :

mateo21@mateo21-desktop:~/tests$ ls -l

total 0

-rw-r--r-- 1 mateo21 mateo21 0 2008-07-31 13:55 one

lrwxrwxrwx 1 mateo21 mateo21 8 2008-07-31 14:15 two -> one

par exemple, si vous supprimezfichier2, il ne se passe rien de mal. Par contre, si vous supprimezfichier1,fichier2pointera vers un fichier qui n'existe plus. Le lien symbolique sera cassé et ne servira donc plus à rien. On parle de « lien mort » ;

* + - * d'autre part, l'avantage des liens symboliques est qu'ils fonctionnent aussi sur des répertoires, contrairement aux liens physiques.

**Les utilisateurs et les droits**

**sudo: exécuter une commande en root**

Le compte user créé par défaut est limité dans ses droits

**L'organisation des utilisateurs sous Linux**

On peut créer autant d'utilisateurs que l'on veut, eux-mêmes répartis dans des groupes.

Il y a un utilisateur « spécial », **root**, aussi appelé superutilisateur. Celui-ci a tous les droits sur la machine.

On ne se connecte en root que très rarement, **lorsque c'est nécessaire**.

un virus ne peut rien faire de plus que vous quand vous êtes connectés avec des droits limités.

Sous Windows, vous êtes toujours connectés en administrateur par défaut (équivalent de root), ce qui explique pourquoi les virus y sont si dangereux.

On peut devenir root **temporairement** à l'aide de la commande sudo.

Cette commande signifie « Faire en se substituant à l'utilisateur » : **S**ubstitute **U**ser **DO**.

Écrivez donc sudo suivi de la commande que vous voulez exécuter, comme ceci :

sudo commande

**sudo su : devenir root et le rester**

Si vous tapez sudo su (tout court), vous passerez root indéfiniment.

Le symbole # à la fin de l'invite de commandes vous indique que vous êtes devenus superutilisateur.

Vous pouvez alors exécuter autant de commandes en root que vous le voulez.

Pour quitter le « mode root », tapez exit (ou faites la combinaison Ctrl + D).

**adduser : gestion des utilisateurs**

adduser et deluser sont des commandes utilisables qu’en mode root. Seul root peut gérer les utilisateurs.

La commande adduser permet d'ajouter un utilisateur. Vous devez au minimum fournir un paramètre : le nom de l'utilisateur à créer.

Pour ajouter Yaya tu devras taper sudo adduser Yaya.

Puis entrer un mot de passe

Le répertoire personnel de patrick est automatiquement créé (/home/patrick) et son compte est préconfiguré.

**passwd : changer le mot de passe**

S'il était nécessaire de changer le mot de passe de patrick par la suite, utilisez la commande passwd en indiquant en paramètre le nom du compte à modifier.

Attention ! Si vous appelez passwd sans préciser de compte en paramètre, c'est le mot de passe de l'utilisateur sous lequel vous êtes connecté que vous changerez ! Ainsi, si vous êtes en root, c'est le mot de passe de root qui sera modifié.

**deluser : supprimer un compte**

Vous pouvez supprimer un compte avec deluser.

deluser patrick

Ne supprimez en aucun cas votre compte utilisateur ! Par exemple, je ne dois surtout pas supprimer le compte mateo21.

En effet, si je le fais, il n'y aura plus que root sur la machine… et Ubuntu interdit de se logger en root. Par conséquent, au prochain démarrage de la machine vous ne pourrez pas vous connecter… et vous serez complètement coincés !

Toutefois, cette commande seule ne supprime pas le répertoire personnel de patrick. Si vous voulez supprimer aussi son home et tous ses fichiers personnels, utilisez le paramètre --remove-home :

deluser --remove-home patrick

adduser et deluser sont des commandes qui n'existent que sous Debian et tous ses descendants, dont Ubuntu.

Partout ailleurs on doit utiliser useradd et userdel, qui sont les commandes Unix traditionnelles fonctionnant partout.

**addgroup : gestion des groupes**

En effet, si vous ne définissez rien, un groupe du même nom que l'utilisateur sera automatiquement créé : ainsi, mateo21 appartient au groupe mateo21 et patrick au groupe patrick.

Afin de savoir à quel groupe appartient un user on fait ls -l

drwx------2 root    root    16384 2007-09-19 18:22 lost+found

drwxr-xrx 65 mateo21 mateo21  4096 2007-11-15 22:40 mateo21

drwxr-xrx  2 patrick patrick  4096 2007-11-15 23:00 patrick

la 3ème colonne indique le propriétaire du fichier ou dossier ; la 4ème indique le groupe qui possède ce fichier ou dossier.

La commande addgroup crée un nouveau groupe. Vous avez juste besoin de spécifier le nom de celui-ci en paramètre :

**usermod : modifier un utilisateur**

La commande usermod permet d'éditer un utilisateur. Elle possède plusieurs paramètres ; nous allons en retenir deux :

* **-l** : renomme l'utilisateur (le nom de son répertoire personnel ne sera pas changé par contre) ;
* **-g** : change de groupe.

Si je veux mettre patrick dans le groupe amis, je ferai donc comme ceci :

usermod -g amis patrick

Et pour remettre patrick dans le groupe patrick comme il l'était avant :

usermod -g patrick patrick

Il est aussi possible de faire en sorte qu'un utilisateur appartienne à plusieurs groupes. Pour ce faire, utilisez le paramètre -G (majuscule).

Exemple : usermod -G amis,paris,collegues patrick.

Séparez les noms des groupes par une virgule, sans espace entre chaque nom de groupe.

Faites très attention en utilisant usermod ! Lorsque vous avez recours à -G, l'utilisateur change de groupe et ce peu importe les groupes auxquels il appartenait auparavant.

Si vous voulez **ajouter** des groupes à un utilisateur (sans perdre les groupes auxquels il appartenait avant cela), utilisez -a :

usermod -aG amis patrick

**delgroup : supprimer un groupe**

Si vous voulez supprimer un groupe, c'est tout simple :

delgroup amis

addgroup et delgroup n'existent que sous Debian et ses dérivés (même remarque que pour adduser et deluser).

Les commandes « traditionnelles » qui fonctionnent partout sont groupadd et groupdel, mais elles offrent moins d'options.

**chown :  : gestion des propriétaires d'un fichier**

Seul l'utilisateur root peut changer le propriétaire d'un fichier.

**chown : changer le propriétaire d'un fichier**

La commande chown, qui doit être utilisée **en tant que root**, attend deux paramètres au moins :

* le nom du nouveau propriétaire ;
* le nom du fichier à modifier.

Cela donne donc :

chown patrick rapport.txt

**chgrp : changer le groupe propriétaire d'un fichier**

chgrp s'utilise exactement de la même manière que chown à la différence près qu'il affecte cette fois le groupe propriétaire d'un fichier.

chgrp amis rapport.txt

Cette commande affectera le fichier rapport.txt au groupe amis.

**chown peut aussi changer le groupe propriétaire d'un fichier !**

Eh oui ! C'est d'ailleurs l'astuce que j'utilise le plus souvent :

chown patrick:amis rapport.txt

Cela affectera le fichier à l'utilisateur patrick et au groupe amis.

Il suffit de séparer par un symbole deux points (« : ») le nom du nouvel utilisateur (à gauche) et le nom du nouveau groupe (à droite).

**chmod : modifier les droits d'accès**

**Le fonctionnement des droits**

Chaque fichier et chaque dossier possède une liste de droits. C'est une liste qui indique qui a le droit de voir le fichier, de le modifier et de l’exécuter.

On peut voir cinq lettres différentes. Voici leur signification :

* **d** (Directory) : indique si l'élément est un dossier ;
* **l** (Link) : indique si l'élément est un lien (raccourci) ;
* **r** (Read) : indique si on peut lire l'élément ;
* **w** (Write) : indique si on peut modifier l'élément ;
* **x** (eXecute) : si c'est un fichier, « x » indique qu'on peut l'exécuter. Ce n'est utile que pour les fichiers exécutables (programmes et scripts).
* Si c'est un dossier, « x » indique qu'on peut le « traverser », c'est-à-dire qu'on peut voir les sous-dossiers qu'il contient si on a le droit de lecture dessus.

Si la lettre apparaît, c'est que le droit existe. S'il y a un tiret à la place, c'est qu'il n'y a aucun droit.

**Nano, l'éditeur de texte du débutant**

**Premiers pas avec Nano**

**Nano est un éditeur de texte, pas un traitement de texte !**

Un **éditeur de texte** est un programme qui permet de modifier des fichiers de texte brut, sans mise en forme (gras, italique, souligné…). Sous Windows, on dispose d'un éditeur de texte très basique : le Bloc-Notes. Sous Linux, on a le choix entre Nano, Vim, Emacs et bien d’autres

Un **traitement de texte** est fait pour rédiger des documents mis en forme. Sous Windows, Word est le plus célèbre traitement de texte ; sous Linux, on possède l'équivalent : Open Office Writer. Ces programmes ne peuvent être utilisés qu'en mode graphique

Pour démarrer le logiciel, il vous suffit simplement de taper nano dans la console :

Le symbole^signifieCtrl(la toucheContrôlede votre clavier). Ainsi, pour quitter Nano, il suffit de taperCtrl + X.

Voici les raccourcis les plus importants :

* Ctrl + G : afficher l'aide ;
* Ctrl + K : couper la ligne de texte (et la mettre dans le presse-papier) ;
* Ctrl + U : coller la ligne de texte que vous venez de couper ;
* Ctrl + C : afficher à quel endroit du fichier votre curseur est positionné (numéro de ligne…) ;
* Ctrl + W : rechercher dans le fichier ;
* Ctrl + O : enregistrer le fichier (écrire) ;
* Ctrl + X : quitter Nano.

**Les paramètres de la commande Nano**

Lorsque vous appelez Nano dans la ligne de commandes, vous pouvez spécifier plusieurs paramètres. Le plus courant est d'indiquer en paramètre le nom du fichier qu'on veut ouvrir. Ainsi :

nano salut.txt

* **-m** : autorise l'utilisation de la souris sous Nano. En console, oui, oui. Vous pouvez vous en servir pour cliquer avec votre souris sur la zone de texte où vous voulez placer votre curseur.
* **-i** : indentation automatique. L'alinéa (tabulations) de la ligne précédente sera respecté lorsque vous irez à la ligne. Très utile lorsque vous éditez un fichier de code source

Si je veux lancer Nano avec toutes ces options à la fois, je peux donc écrire :

nano -miA salut.txt

**Configurer Nano avec .nanorc**

il existe un fichier de configuration de Nano qui indique toutes vos préférences. Celui-ci s’appelle.nanorc.

La plupart des fichiers de configuration commencent par un point. Cela permet de « cacher » le fichier quand on fait unls. Bien entendu, comme vous devriez maintenant le savoir, les fichiers cachés peuvent toujours être affichés en utilisant le paramètre-a :ls -a.

Ce fichier est lu par Nano à chaque fois que vous le démarrez.

Si l fichier n’existe pas alors nano sera utilisé avec les options par défaut, sinon on créé le fichier en tapant : nano .nanorc

Dans ce fichier, vous devez écrire une commande par ligne.

Chaque commande commence par un **set** (pour activer) ou un **unset** (pour désactiver) suivi de l'option qui vous intéresse.

Par exemple, pour activer la souris, écrivez : set mouse

**Le nanorc global et la coloration syntaxique**

Ce fichier.nanorc dans votre home est très pratique car il vous permet de définir vos propres options. Mais si vous avez dix utilisateurs sur votre machine et que vous voulez activer le support de la souris pour tout le monde. Il faudra modifier le fichier nanorc« global » qui est pris en compte pour tout le monde. Celui-ci est situé dans/etc/nanorc

Ce fichier ne peut être modifié que par root. Je vous conseille donc de l'ouvrir avec un sudo(ou dans une console en root si vous avez fait sudo su avant) :

sudo nano /etc/nanorc

Toutes les options disponibles sont présentes mais en mode commentaires, il suffit de décommenter celles que l’on souhaite utiliser.

**Configurer sa console avec .bashrc**

C’est le fichier de configuration de l'ensemble de la console.Il se situe dans votre répertoire personnel et celui-ci existe déjà normalement.

On peut personnaliser l’invite de commande.

**Créer des alias**

Les alias sont des commandes que vous créez et qui sont automatiquement transformées en d'autres commandes.

Descendez un peu plus bas dans le fichier, vous trouverez des lignes commentées commençant par « alias ».

Attention à ne pas mettre d'espace autour du symbole « = ».

**Édition du bashrc global**

Si vous voulez définir des alias ou modifier l'invite de commandes pour tous vos utilisateurs, vous pouvez le faire en une seule fois en éditant le fichier bashrc global situé dans :/etc/bash.bashrc.

Ce bashrc doit être édité en root.

Les éléments du bashrc personnel ont la priorité sur ceux du bashrc global. Si un même alias est défini dans les deux, c'est celui du bashrc personnel qui sera pris en compte.

**Et aussi… le.profile**

De même qu'il existe un~/.bashrc et un/etc/bash.bashrc, il existe un~/.profile et un/etc/profile. Quelle est la différence ?

En gros, le **.profile** est lu à chaque nouvelle console dans laquelle vous vous loggez (vous rentrez votre login / mot de passe). C'est le cas des consoles que vous ouvrez avecCtrl + Alt + F1àF6(tty1àtty6).

Le **.bashrc** est lu lorsque vous ouvrez une console dans laquelle vous ne vous loggez pas. C'est le cas des consoles que vous ouvrez en mode graphique (Terminal sous Unity, Konsole sous KDE).

C'est un peu compliqué pour pas grand-chose au final. Dans la pratique, le.profilefait par défaut appel au.bashrc… Donc il suffit d'éditer votre.bashrcet vous modifierez ainsi les options de toutes vos consoles : celles avec et sans login.

**Installer des programmes avec apt-get**

**RTFM : lisez le manuel !**

RTMF est l’abréviation de Read the fucking manual

**man : afficher le manuel d'une commande**

La commande magique que nous allons expérimenter tout au long de ce chapitre est man, qui est l'abréviation de **man**ual.

La commande man s'utilise très simplement : elle prend en paramètre le nom de la commande dont vous voulez lire la doc.

Par exemple, si je veux tout savoir sur la commande mkdir, je dois écrire : man mkdir

**Se déplacer dans le manuel**

On peut se déplacer à l’aide des flèches directionnelles.

Appuyez sur la touche Home (aussi appelée Origine) pour revenir au début du manuel, et sur Fin pour aller à la fin.

Appuyez sur la touche / (slash) pour effectuer une recherche ; c'est très pratique ! Tapez ensuite le mot que vous recherchez dans le manuel puis appuyez sur Entrée.

Appuyez sur la touche Q pour quitter le manuel à tout moment

**Les principales sections du manuel**

* NAME : le nom de la commande dont vous êtes en train d'afficher le manuel ainsi qu'une courte description de son utilité.
* SYNOPSIS : c'est la liste de toutes les façons d'utiliser la commande.
* DESCRIPTION : une description plus approfondie de ce que fait la commande. On y trouve aussi la liste des paramètres et leur signification. C'est en général la section la plus longue.

En installant manpages-fr tu peux installer un manuel en français.

La section SYNOPSIS de mkdir est présentée sur la figure suivante.



Détaillons point par point ce SYNOPSIS.

* mkdir : pour utiliser la commande mkdir, vous devez commencer par taper mkdir ; ça, c'est logique.
* [OPTION] : après mkdir, vous pouvez écrire une option. Dans le SYNOPSIS, on met des crochets pour indiquer que c'est facultatif. Vous n'êtes donc pas obligés d'écrire une option.
* DIRECTORY : c'est le nom du répertoire à créer. Ce paramètre est obligatoire puisqu'il n'est pas entre crochets.

Les points de suspension après DIRECTORY nous indiquent qu'on peut répéter le nom du répertoire autant de fois que l'on veut, ce qui nous permet d'en créer plusieurs d'un coup.

On peut donc aussi utiliser mkdir comme ceci :

mkdir images videos musiques

… ce qui aura pour effet de créer trois dossiers : images, videos et musiques.

Le mode -v (verbeux) demande à la commande de détailler ce qu'elle fait.

Le mode -i demande la confirmation avant de faire l’action.

**apropos : trouver une commande**

La commande apropos est donc un peu l'inverse de man : elle vous permet de retrouver une commande.

Prenons un exemple : vous recherchez une commande (que vous avez installée) en rapport avec le son parce que vous aimeriez bien savoir comment modifier le volume en console.

Vous pouvez taper :

apropos sound

… ce qui va rechercher toutes les commandes qui parlent de son (sound) dans leur page du manuel.

**D'autres façons de lire le manuel**

La plupart des commandes acceptent un paramètre -h (et parfois son équivalent plus long --help) qui provoque l'affichage d'une aide résumée. Parfois cette aide est d'ailleurs plus facile à lire que celle du man, ce qui fait qu'il m'arrive de l'utiliser de temps en temps.

Par exemple :

apt-get -h

Parfois, il n'y a pas de page de manuel pour une commande (man ne fonctionne pas pour cette dernière) mais le -h ou le --help fonctionne. Pensez-y !

**La commande whatis**

La commande whatis est une sorte de man très allégé. Elle donne juste l'en-tête du manuel pour expliquer en deux mots à quoi sert la commande

**Extraire, trier et filtrer des données**

**grep : filtrer des données**

La commande grep est essentielle.

Son rôle est de rechercher un mot dans un fichier et d'afficher les lignes dans lesquelles ce mot a été trouvé.

L'avantage de cette commande est qu'elle peut être utilisée de manière très simple ou plus complexe (mais plus précise) selon les besoins en faisant appel aux expressions régulières.

Les expressions régulières constituent un moyen très puissant de rechercher un texte. On les utilise non seulement dans la ligne de commandes Linux, mais aussi dans des éditeurs de texte avancés et dans de nombreux langages de programmation tels que PHP.

Utiliser grep simplement

grep texte nomfichier

Le premier paramètre est le texte à rechercher, le second est le nom du fichier dans lequel ce texte doit être recherché.

Essayons par exemple de rechercher le mot « alias » dans notre fichier de configuration.bashrc

grep alias .bashrc

Notez qu'il n'est pas nécessaire de mettre des guillemets autour du mot à trouver, sauf si vous recherchez une suite de plusieurs mots séparés par des espaces, comme ceci :

grep "Site du Zéro" monfichier

-n : connaître les numéros des lignes

Vous pouvez afficher les numéros des lignes retournées avec-n :

-v : inverser la recherche : ignorer un mot

Si, au contraire, vous voulez connaître toutes les lignes qui **ne contiennent pas** un mot donné, utilisez-v :

$ grep -v alias .bashrc

Cette fois, on récupère toutes les lignes du fichier.bashrcqui ne contiennent pas le mot « alias ».

-r : rechercher dans tous les fichiers et sous-dossiers

Si vous ne savez pas dans quel fichier se trouve le texte que vous recherchez, vous pouvez sortir l'artillerie lourde : l'option-r(*recursive*). Cette fois, il faudra indiquer en dernier paramètre le **nom du répertoire** dans lequel la recherche doit être faite (et non pas le nom d'un fichier).

grep -r "Site du Zéro" code/

Utiliser grep avec des expressions régulières

Pour faire des recherches plus poussées – pour ne pas dire des recherches *très poussées* –, vous devez faire appel aux expressions régulières. C'est un ensemble de symboles qui va vous permettre de dire à l'ordinateur très précisément ce que vous recherchez.

|  |  |
| --- | --- |
| **Caractère spécial** | **Signification** |
| . | Caractère quelconque |
| ^ | Début de ligne |
| $ | Fin de ligne |
| [] | Un des caractères entre les crochets |
| ? | L'élément précédent est optionnel (peut être présent 0 ou 1 fois) |
| \* | L'élément précédent peut être présent 0, 1 ou plusieurs fois |
| + | L'élément précédent doit être présent 1 ou plusieurs fois |
| | | Ou |
| () | Groupement d'expressions |

Rechercher toutes les lignes comprenant le mot Alias dans le fichier bashrc:

grep Alias .bashrc

Rechercher toutes les lignes commençant par le mot Alias dans le fichier bashrc:

grep ^Alias .bashrc

Rechercher toutes les lignes finissant par le mot vie dans le fichier bashrc :

grep $vie .bashrc

Rechercher toutes les lignes qui contiennent un nombre compris entre 0 et 4.

grep  [0-4] .bashrc

Renvoie toutes les lignes qui contiennent un caractère alphabétique compris entre « a » et « z » ou entre « A » et « Z »

grep  [a-zA-Z] .bashrc

**sort : trier les lignes**

On utilise la commande sort lorsqu'on a besoin de trier le contenu d'un fichier.

Syntaxe : sort nomFichier

Si tu as un fichier avec des noms, ça va les trier par ordre alphabétique.

**-o : écrire le résultat dans un fichier**

Syntaxe: sort -o newFichier fichierTrié

**-r : trier en ordre inverse**

**-n : trier des nombres**

Syntaxe : sort -n nomFichier

**wc : compter le nombre de lignes**

La commandewcsignifie word count. C'est donc a priori un compteur de mots mais en fait, on lui trouve plusieurs autres utilités : compter le nombre de lignes (très fréquent) et compter le nombre de caractères.

Comme les précédentes, la commande wc travaille sur un fichier.

Syntaxe: wc noms.txt

Va afficher par exemple: 8 8 555

Ces trois nombres signifient, dans l'ordre :

1. le nombre de lignes.
2. le nombre de mots.
3. le nombre d’octets.

Quelques paramètres :

**-l : compter le nombre de lignes**

Pour avoir uniquement le nombre de lignes, utilisez-l :

Syntaxe: wc -l nomFichier

**-w : compter le nombre de mots**

**-m : compter le nombre de caractères**

**uniq : supprimer les doublons**

Syntaxe: uniq nomFichier

N'oubliez pas qu'il faut que le fichier soit trié pour que la commande uniq fonctionne !

**cut : couper une partie du fichier**

**Couper selon le nombre de caractères**

Par exemple, si vous souhaitez conserver uniquement les caractères 2 à 5 de chaque ligne du fichier, vous taperez :

cut -c 2-5 noms.txt

Pour conserver du 1er au 3ème caractère :

$ cut -c -3 noms.txt

Comme vous pouvez le voir, si on ne met pas de chiffre au début,cutcomprend que vous voulez parler du premier caractère.

De même, pour conserver du 3ème au dernier caractère :

$ cut -c 3- noms.txt

**Couper selon un délimiteur**

Imagine que tu as un fichier notes.csv ainsi :

Fabrice,18 / 20,Excellent travail

Mathieu,3 / 20,Nul comme d'hab

Sophie,14 / 20,En nette progression

Mélanie,9 / 20,Allez presque la moyenne !

Corentin,11 / 20,Pas mal mais peut mieux faire

Albert,20 / 20,Toujours parfait

Benoît,5 / 20,En grave chute

Comme le nom CSV l'indique, les virgules servent à séparer les colonnes. Ces dernières contiennent, dans l'ordre :

* le prénom ;
* la note ;
* un commentaire.

Imaginons que nous souhaitions extraire de ce fichier la liste des prénoms. Comment nous y prendrions-nous ?

Vous allez avoir besoin d'utiliser deux paramètres :

* -d : indique quel est le délimiteur dans le fichier ;
* -f : indique le numéro du ou des champs à couper.

$ cut -d , -f 1 notes.csv

Fabrice

Vincent

Sophie

Mélanie

Corentin

Albert

Benoît

Pour avoir les champs n°1 et n°3 (le prénom et le commentaire) :

$ cut -d , -f 1,3 notes.csv

De même si on veut conserver tous les champs 2, 3 et 4, il faudra écrire :

$ cut -d , -f 2-4 notes.csv

D'autre part,cut -d , -f 3- notes.csvconserve les champs du n°3 jusqu'à la fin.

**Les flux de redirection**

Quand tu tapes ls, le résultat est affiché sur la console.

Mais il est possible de le rediriger ailleurs : dans un fichier ou en entrée d'une autre commande pour effectuer des « chaînes de commandes » Ainsi, le résultat d'une commande peut en déclencher une autre !

À l'aide de petits symboles spéciaux, appelés flux de redirection

**> et >> : rediriger le résultat dans un fichier**

**> : rediriger dans un nouveau fichier**

Prenons le fichier suivant famille créé avec nano :

Qeyss, 40, Informaticien

Saylmane, 20, Agent d’accueil

Souheyb, 16, Etudiant

Je peux afficher sur la console que les noms avec la commande suivante : cut -d , -f 1 famille

Si je tape : cut -d , -f 1 famille > noms

Cela va créer un fichier noms qui comprendra le résultat de la commande à savoir tout ce qu’il y a dans la première colonne, c’est à dire les noms.

Attention : si le fichier existait déjà il sera écrasé sans demande de confirmation !

**>> : rediriger à la fin d'un fichier**

Le double chevron >> sert lui aussi à rediriger le résultat dans un fichier, mais cette fois à la fin de ce fichier.

Avantage : vous ne risquez pas d'écraser le fichier s'il existe déjà. Si le fichier n'existe pas, il sera créé automatiquement.

**2>, 2>> et 2>&1 : rediriger les erreurs**

Il faut savoir que toutes les commandes produisent deux flux de données différents, comme le montre la figure suivante :

* **la sortie standard** : pour tous les messages (sauf les erreurs) ;
* **la sortie d'erreurs** : pour toutes les erreurs.

Par défaut, tout s'affiche dans la console : la sortie standard comme la sortie d’erreurs.

Prenons un exemple concret pour voir comment ça se passe.

Supposons que vous fassiez un cat d’un fichier pour afficher son contenu. Il y a deux possibilités :

* **si tout va bien**, le résultat (le contenu du fichier) s'affiche sur la sortie standard ;
* **s'il y a une erreur**, celle-ci s'affiche dans la sortie d'erreurs.

**Rediriger les erreurs dans un fichier à part**

On pourrait souhaiter enregistrer les erreurs dans un fichier à part pour ne pas les oublier et pour pouvoir les analyser ensuite.

Pour cela, on utilise l'opérateur 2>. Vous avez bien lu : c'est le chiffre 2 collé au chevron que nous avons utilisé tout à l’heure.

Faisons une seconde redirection à la fin de cette commande cut :

cut -d , -f 1 fichier\_inexistant.csv > eleves.txt 2> erreurs.log

Il y a deux redirections ici :

* > eleves.txt : redirige le résultat de la commande (sauf les erreurs) dans le fichier eleves.txt. C'est la sortie standard ;
* 2> erreurs.log : redirige les erreurs éventuelles dans le fichier erreurs.log. C'est la sortie d'erreurs.

Vous pouvez vérifier : si fichier\_inexistant.csv n'a pas été trouvé, l'erreur aura été inscrite dans le fichier erreurs.log au lieu d'être affichée dans la console.

Notez qu'il est aussi possible d'utiliser 2>>

pour ajouter les erreurs à la fin du fichier.

**Fusionner les sorties**

Parfois, on n'a pas envie de séparer les informations dans deux fichiers différents. Heureusement, il est possible de fusionner les sorties dans un seul et même fichier. Comment ?

Il faut utiliser le code suivant : 2>&1.

Cela a pour effet de rediriger toute la sortie d'erreurs dans la sortie standard. Traduction pour l'ordinateur : « envoie les erreurs au même endroit que le reste ».

Essayez donc ceci :

cut -d , -f 1 fichier\_inexistant.csv > eleves.txt 2>&1

Tout ira désormais dans eleves.txt : le résultat (si cela a fonctionné), de même que les erreurs (s'il y a eu un problème).

**< et << : lire depuis un fichier ou le clavier**

**< : lire depuis un fichier**

Écrire cat < notes.csv est strictement identique au fait d'écrire cat notes.csv… du moins en apparence. Le résultat produit est le même, mais ce qui se passe derrière est très différent.

* Si vous écrivez cat notes.csv, la commande cat reçoit en entrée le nom du fichier notes.csv qu'elle doit ensuite se charger d'ouvrir pour afficher son contenu.
* Si vous écrivez cat < notes.csv, la commande cat reçoit **le contenu** de notes.csv qu'elle se contente simplement d'afficher dans la console. C'est le shell (le programme qui gère la console) qui se charge d'envoyer le contenu de notes.csv à la commande cat.

Bref, ce sont deux façons de faire la même chose mais de manière très différente.

**| : chaîner les commandes**

le **pipe**| permet de chaîner des commandes.

Le pipe | n'est pas un symbole qu'on a l'habitude d'écrire. Pourtant, il y en a forcément un sur votre clavier (parfois représenté sous la forme d'une ligne verticale en pointillés).

Sur un clavier AZERTY français par exemple, vous pouvez l'écrire en combinant les touches Alt Gr + 6 et sur un clavier belge, Alt Gr + 1. Sur un clavier Mac, c'est Alt + Shift + L

**Surveillez l’activité du système**

**Exécuter des programmes en arrière-plan**

**Exécuter un programme à une heure différée**

cc

**Connexion sécurisée à distance avec SSH**

Cc

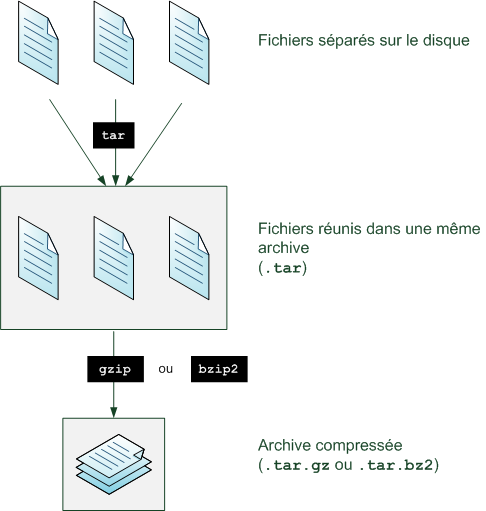
**Archiver et compresser**

On utilise gzip et bzip2 Toutefois, contrairement à zip et rar, le gzip et le bzip2ne sont capables de compresser qu'un seul fichier à la fois et ne peuvent donc pas créer un « paquetage » de plusieurs fichiers.

Tar : assembler des fichiers dans une archive

Sous Linux, on a depuis longtemps pris l'habitude de procéder en deux étapes :

1. réunir les fichiers dans un seul gros fichier appelé **archive**. On utilise pour cela le programme tar ;
2. compresser le gros fichier ainsi obtenu à l'aide de gzip ou de bzip2.



Les formats zip et rar ne séparent pas les étapes comme sur le schéma suivant. Ils sont capables d'assembler plusieurs fichiers en une archive et de la compresser en même temps.

Certains formats de fichier sont déjà compressés. C'est le cas des images jpeg, png et gif, mais aussi de la plupart des vidéos. Vous pouvez bien entendu les assembler dans une archive tar et même les compresser, mais vous ne les rendrez en général pas plus petits car ils ont déjà été compressés.

Tu crée un dossier dans lequel tu vas mettre tous les fichiers que tu veux compresser, puis tu dois créer une archive tar :

tar -cvf nom\_archive.tar nom\_dossier/

J'utilise trois options :

* -c : signifie créer une archive tar ;
* -v : signifie afficher le détail des opérations ;
* -f : signifie assembler l'archive dans un fichier.

#### -tf : afficher le contenu de l'archive sans l'extraire

Vous venez de recevoir une archive tar qu'on vous a envoyée. Bien. Mais que contient-elle ?  
Avant d'extraire quoi que ce soit, vous aimeriez peut-être voir son contenu. C'est possible avec -tf :

$ tar -tf tutoriels.tar

#### -rvf : ajouter un fichier

Si vous avez oublié un fichier, vous pouvez toujours l'ajouter par la suite avec -rvf :

$ tar -rvf tutoriels.tar fichier\_supplementaire.tuto

tutoriels/fichier\_supplementaire.tuto

#### -xvf : extraire les fichiers de l'archive

Pour extraire les fichiers, on va utiliser les options -xvf (-x pour eXtract) :

$ tar -xvf tutoriels.tar

contenu de l'archive sans l'extraire

* gzip : c'est le plus connu et le plus utilisé ;
* bzip2 : il est un peu moins fréquemment utilisé. Il compresse mieux mais plus lentement que gzip.

Ces programmes sont simples à utiliser. Ils prennent comme paramètre le nom du fichier à compresser. Ils le compressent et modifient ensuite son nom.  
Concrètement, ils ajoutent un suffixe pour indiquer que l'archive a été compressée :

* .tar.gz : si l'archive a été compressée avec gzip ;
* .tar.bz2 : si l'archive a été compressée avec bzip2.

#### gzip : la compression la plus courante

Concrètement, le programme gzip s'utilise de la manière la plus simple qui soit :

gzip tutoriels.tar

L'archive est compressée et gagne ensuite le suffixe .gz. Elle s'appelle donc désormais tutoriels.tar.gz. Voilà pourquoi vous voyez circuler sur l'internet des fichiers .tar.gz : cela signifie que ce sont des archives compressées !

Pour décompresser l'archive ensuite, il suffit d'utiliser gunzip :

gunzip tutoriels.tar.gz

L'archive retrouve son état non compressé en .tar. Vous pouvez maintenant extraire les fichiers de l'archive

Archiver et compresser en même temps

##### -zcvf : archiver et compresser en gzip

Vous connaissez tar -cvf pour créer une archive tar. Si vous rajoutez l'option -z, l'archive sera automatiquement compressée avec gzip.

tar -zcvf tutoriels.tar.gz tutoriels/

Unzip et unrar pour décompresser les .zip et .rar

#### unzip : décompresser un .zip

Vous venez de recevoir un .zip ?  
Pas de panique ! Le programme unzip est capable de l'extraire. Il est peut-être installé par défaut, mais si vous ne l'avez pas, vous savez ce qu'il vous reste à faire :

sudo apt-get install unzip

Ceci étant fait, l'utilisation d'unzip est très simple :

unzip archive.zip

#### unrar : décompresser un .rar

Il vous faut installer le paquet unrar pour pouvoir utiliser cette commande :

sudo apt-get install unrar

Ensuite, pour extraire :

unrar e tutoriels.rar

#### En résumé

* Pour regrouper plusieurs fichiers et dossiers au sein d'un même fichier (appelé archive), on utilise le programme tar. Celui-ci ne compresse pas les fichiers par défaut, contrairement à zip.
* Il est possible de compresser une archive tar avec le programme gzip (très couramment utilisé) ou bzip2 (meilleure compression mais plus lente).
* Les archives non compressées ont l'extension .tar, les archives compressées ont l'extension .tar.gz (pour gzip) ou .tar.bz2 (pour bzip2).
* On utilise peu les formats de compression zip et rar sous Linux, mais il est possible de décompresser ces types de fichiers avec les programmes unzip et unrar. Ceux-ci ne sont en général pas installés par défaut.

Analyser le réseau et filtrer le trafic avec un pare-feu

**Host & whois**

**Ifconfig et netstat**

**Iptables**

xx

Cc

c

c

**Introduction aux scripts shell**

Langage de programmation Shell

C’est un minilangage de programmation intégré à Linux. Il permet d'automatiser la plupart de vos tâches : sauvegarde des données, surveillance de la charge de votre machine, etc.

**Qu'est-ce qu'un shell ?**

C’est un environnement console, c’est l’interpréteur de commande, il fait le lien entre l’utilisateur et le système.

Voici les noms de quelques-uns des principaux shells qui existent.

* **sh** : Bourne Shell. L'ancêtre de tous les shells.
* **bash** : Bourne Again Shell. Une amélioration du Bourne Shell, disponible par défaut sous Linux et Mac OS X.
* **ksh** : Korn Shell. Un shell puissant assez présent sur les Unix propriétaires, mais aussi disponible en version libre, compatible avec bash.
* **csh** : C Shell. Un shell utilisant une syntaxe proche du langage C.
* **tcsh** : Tenex C Shell. Amélioration du C Shell.
* **zsh** : Z Shell. Shell assez récent reprenant les meilleures idées de bash, ksh et tcsh.

L’ancêtre de tous les shells est le sh (Bourne Shell). C'est le plus vieux et il est installé sur tous les OS basés sur Unix. Il est néanmoins pauvre en fonctionnalités par rapport aux autres shells.

Le bash (Bourne Again Shell) est le shell par défaut de la plupart des distributions Linux mais aussi celui du terminal de Mac OS X.

sh reste toujours plus répandu que bash. En fait, vous pouvez être sûrs que tous les OS basés sur Unix possèdent sh, mais ils n'ont pas tous forcément bash.

**À quoi sert un shell ?**

Le shell est le programme qui gère l'invite de commandes. C'est donc le programme qui attend que vous rentriez des commandes.

Un compilateur prend le fichier source et le compile en un fichier binaire compréhensible par le noyau de l’OS. Le Shell lui interprète et exécute directement les instructions.

Le .bashrc est le fichier de configuration du bash que Linux vous fait utiliser par défaut. Chaque personne peut avoir son .bashrc pour personnaliser son invite de commandes, ses alias, etc.

**Installer un nouveau shell**

apt-get install ksh

Une fois installé, il faut demander à l'utiliser pour votre compte utilisateur. Pour cela, tapez :

$ chsh

chsh signifie Change Shell.

On vous demandera où se trouve le programme qui gère le shell. Vous devrez indiquer /bin/ksh pour ksh, /bin/sh pour sh, /bin/bash pour bash, etc.

Un script shell **dépend** d'un shell précis. En gros, le langage n'est pas tout à fait le même selon que vous utilisez sh, bash, ksh, etc.

Le bash est un bon compromis entre sh (le plus compatible) et ksh / zsh (plus puissants).

**Notre premier script**

Avec Vim on va créer un nouveau fichier en lui donnant l’extension .sh pour indiquer que c’est un script shell. Cette extension n’est pas obligatoire.

**Indiquer le nom du shell utilisé par le script**

La première chose à faire dans un script shell est d'indiquer… quel shell est utilisé.

Le #! est appelé le **sha-bang**.

Bien que non indispensable, cette ligne permet de s'assurer que le script est bien exécuté avec le bon shell.

En l'absence de cette ligne, c'est le shell de l'utilisateur qui sera chargé. Cela pose un problème : si votre script est écrit pour bash et que la personne qui l'exécute utilise ksh, il y a de fortes chances pour que le script ne fonctionne pas correctement !

Allez, on va commencer par quelque chose de très simple : un ls. On va donc créer un script bash qui va juste se contenter d'afficher le contenu du dossier courant !/bin/bash

ls

**Les commentaires**

Notez que vous pouvez aussi ajouter des commentaires dans votre script. Ce sont des lignes qui ne seront pas exécutées mais qui permettent d'expliquer ce que fait votre script.

Tous les commentaires commencent par un #

**Exécuter le script bash**

**Donner les droits d'exécution au script**

Si vous faites un ls –l pour voir votre fichier qui vient d'être créé, vous verrez qu’il n’y a pas le droit d’exécution.

Or, pour exécuter un script, il faut que le fichier ait le droit « exécutable ». Le plus simple pour donner ce droit est d'écrire :

$ chmod +x essai.sh

**Exécution du script**

Le script s'exécute maintenant comme n'importe quel programme, en tapant « ./ » devant le nom du script :

$ ./essai.sh

Que fait le script ? Il fait juste un ls, donc il affiche la liste des fichiers présents dans le répertoire

Vous pouvez déjà modifier votre script pour qu'avant toute chose il vous donne également le nom du répertoire dans lequel vous vous trouvez :

#!/bin/bash

pwd

ls

**Exécution de débogage**

Il faut l'exécuter comme ceci :

$ bash -x essai.sh

On appelle en fait directement le programme bash et on lui ajoute en paramètre un -x (pour lancer le mode débogage) ainsi que le nom de notre script à déboguer.

Le shell affiche alors le détail de l'exécution de notre script, ce qui peut nous aider à retrouver la cause de nos erreurs.

**Créer sa propre commande**

Actuellement, le script doit être lancé via ./essai.sh et vous devez être dans le bon répertoire.

Ou alors vous devez taper le chemin en entier, comme

/home/qeyss/scripts/essai.sh

Afin de pouvoir l’executer depuis n’importe quel endroit sans avoir à taper ./ devant il faut placer le script dans un des répertoires du PATH. Le PATH est une variable système qui indique où sont les programmes exécutables sur votre ordinateur. Si vous tapez echo $PATH vous aurez la liste de ces répertoires « spéciaux ».

Il vous suffit donc de déplacer ou copier votre script dans un de ces répertoires, comme /bin ou /usr/bin ou /usr/local/bin (ou encore un autre répertoire du PATH). Notez qu'il faut être root pour pouvoir faire cela

Une fois que c'est fait, vous pourrez alors taper simplement essai.sh pour exécuter votre programme et ce quel que soit le répertoire dans lequel vous vous trouverez !

Afficher et manipuler des variables

Il ne faut pas mettre d’espace avant ou après le signe =

age=25

nom=”qeyss”

**echo : Afficher une variable**

On utilise echo pour afficher une variable

En tapant echo Hi guys , ça affichera Hi guys, tout ce qui vient après est considéré comme des paramètres à afficher.

Pour afficher une variable, nous allons de nouveau utiliser son nom précédé du symbole dollar $

echo $nom

Il faut utiliser les doubles quotes pour afficher la valeur d’une variable, ex :

message=' coucou '

Echo " Il a dit : $message "

##### Les back quotes ` `

Un peu particulières, les back quotes demandent à bash d'**exécuter** ce qui se trouve à l'intérieur.

message=`pwd`

echo "Vous êtes dans le dossier $message"

Vous êtes dans le dossier /home/mateo21/bin

La commande pwd a été exécutée et son contenu inséré dans la variable message ! Nous avons ensuite affiché le contenu de la variable.

Read : demander une saisie

Syntaxe : read NomVariable

#!/bin/bash

read nom

echo "Bonjour $nom !"

Là rien ne va s’afficher car shell attend que tu tapes ton nom, c’est comme un input en python

Mathieu

Bonjour Mathieu !

Tu peux afficher plusieurs variables :

#!/bin/bash

read nom prenom

echo "Bonjour $nom $prenom !"

#### -p : afficher un message de prompt

Avec l'option -p de read, vous pouvez faire cela :

#!/bin/bash

read -p 'Entrez votre nom : ' nom

echo "Bonjour $nom !"

Notez que le message 'Entrez votre nom' a été entouré de quotes. Si on ne l'avait pas fait, le bash aurait considéré que chaque mot était un paramètre différent !

Résultat :

Entrez votre nom : Mathieu

Bonjour Mathieu !

#### -s : ne pas afficher le texte saisi

 le paramètre -s masque les caractères que vous saisissez. Cela vous servira notamment si vous souhaitez que l'utilisateur entre un mot de passe :

#!/bin/bash

read -p 'Entrez votre mot de passe : ' -s pass

echo -e "\nMerci ! Je vais dire à tout le monde que votre mot de passe est $pass ! :)"

Entrez votre mot de passe :

Merci ! Je vais dire à tout le monde que votre mot de passe est supertopsecret38 ! :)

omme un input en python

Les variables d’environnement

Les tableaux

Pour définir un tableau, on peut faire comme ceci :

tableau=('valeur0' 'valeur1' 'valeur2')

Cela crée une variable tableau qui contient trois valeurs (valeur0, valeur1, valeur2).

Pour accéder à une case du tableau, il faut utiliser la syntaxe suivante :

${tableau[2]}

Les conditions

La syntaxe en bash est la suivante :

if [ test ]

then

        echo "C'est vrai"

fi

Vous noterez — c'est très important — qu'il y a des espaces à l'intérieur des crochets. On ne doit pas écrire[test]mais[ test ] !

#!/bin/bash

nom="Bruno"

if [ $nom = "Bruno" ]

then

        echo "Salut Bruno !"

fi

Sinon

#!/bin/bash

if [ $1 = "Bruno" ]

then

        echo "Salut Bruno !"

else

        echo "J'te connais pas, ouste !"

fi

Sinon si

#!/bin/bash

if [ $1 = "Bruno" ]

then

        echo "Salut Bruno !"

elif [ $1 = "Michel" ]

then

        echo "Bien le bonjour Michel"

elif [ $1 = "Jean" ]

then

        echo "Hé Jean, ça va ?"

else

        echo "J'te connais pas, ouste !"

fi

Opérateurs :

-lt = <

-le = <=

-gt = >

-ge = >=

#!/bin/bash

if [ $1 -ge 20 ]

then

        echo "Vous avez envoyé 20 ou plus"

else

        echo "Vous avez envoyé moins de 20"

fi

#### Effectuer plusieurs tests à la fois

Dans unif, il est possible de faire plusieurs tests à la fois. En général, on vérifie :

* si un test est vrai **ET** qu'un autre test est vrai ;
* si un test est vrai **OU** qu'un autre test est vrai.

Les deux symboles à connaître sont :

* **&&** : signifie « et » ;
* **||** : signifie « ou ».

#!/bin/bash

if [ $# -ge 1 ] && [ $1 = 'koala' ]

then

        echo "Bravo !"

        echo "Vous connaissez le mot de passe"

else

        echo "Vous n'avez pas le bon mot de passe"

fi

Case : tester plusieurs conditionsà la fois :

#!/bin/bash

case $1 in

        "Bruno")

                echo "Salut Bruno !"

                ;;

        "Michel")

                echo "Bien le bonjour Michel"

                ;;

        "Jean")

                echo "Hé Jean, ça va ?"

                ;;

        \*)

                echo "J'te connais pas, ouste !"

                ;;

esac

Les boucles :

While

while [ test ]

do

        echo 'Action en boucle'

done

For:

#!/bin/bash

for variable in 'valeur1' 'valeur2' 'valeur3'

do

        echo "La variable vaut $variable"

done

!/bin/bash

for animal in 'chien' 'souris' 'moineau'

do

        echo "Animal en cours d'analyse : $animal"

done

Animal en cours d'analyse : chien

Animal en cours d'analyse : souris

Animal en cours d'analyse : moineau

omme un input en python

o